Gleason

Minh Chau Do

09 10 2023

Daten von Gleason

```r
#Datensortierung

# In Mortalität umrechnen
gleason_r$V2 <- 100 - gleason_r$V2

gleason_r <- gleason_r[complete.cases(gleason_r),]

gleason_r <- gleason_r[order(gleason_r$V2), ]

xr_gleason <- sort(gleason_r$V1)
yr_gleason <- gleason_r$V2



# In Mortalität umrechnen
gleason_b$V2 <- 100 - gleason_b$V2

gleason_b <- gleason_b[complete.cases(gleason_b),]

gleason_b <- gleason_b[order(gleason_b$V2), ]

xb_gleason <- sort(gleason_b$V1)
yb_gleason <- gleason_b$V2
```

Startfunktion

Um die bestmögliche Anpassung an den vorliegenden Datenpunkten darzustellen wurde mithilfe einer Funktion der beste Startparameter mit dem kleinsten Fehler ermittelt, da die Anpassung stark von den initialen Startwerten abhängig ist.

```r
find_best_start_2parameter <- function(p, q, max_beta, max_eta, steps_beta,
                                       steps_eta, fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
  best_starts <- matrix(nrow = 0, ncol = 2)  # Matrix für Startparameter

  for (beta in seq(0, max_beta, by = steps_beta)) {
    for (eta in seq(0, max_eta, by = steps_eta)) {
      start_params <- c(beta, eta)
```

```
      # Schätze die Parameter mit den aktuellen Startparametern
      if(identical(as.character(substitute(fitting_function)), "getstuexp2")){
        output <- capture.output({
        result <- getstuexp2(
                        p = p, q = q, start = start_params,
                        show.output = TRUE, plot = FALSE, wert1 = 2
                      )
        })
      }
      else{
        output <- capture.output({
        result <- fitting_function(p = p, q = q, start = start_params,
                              show.output = TRUE, plot = FALSE)
        })
      }

      # Berechne den Fehler (gleason_r_1$value) für die aktuellen Startparameter
      current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

      # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
      if (!is.na(current_error)) {
        best_errors <- c(best_errors, current_error)
        best_starts <- rbind(best_starts, start_params)
      }
    }
  }


  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}


find_best_start_3parameter <- function(p, q, max_shape1 = 10, max_shape2 = 10,
                                       max_scale = 10, steps_shape1, steps_shape2,
                                       steps_scale, fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
  best_starts <- matrix(nrow = 0, ncol = 3)  # Matrix für Startparameter

  for (shape1 in seq(0, max_shape1, by = steps_shape1)) {
    for (shape2 in seq(0, max_shape2, by = steps_shape2)) {
      for (scale in seq(0, max_shape1, by = steps_scale)) {
        start_params <- c(shape1, shape2, scale)
```

```
        # Schätze die Parameter mit den aktuellen Startparametern
        if(identical(as.character(substitute(fitting_function)), "getstuexp3")){
          output <- capture.output({
            result <- getstuexp3(
                          p = p, q = q, start = start_params,
                          show.output = TRUE, plot = FALSE, wert1 = 2, wert2 = 6)
          })
      }

        else{
          output <- capture.output({
            result <- fitting_function(p = p, q = q, start = start_params,
                                 show.output = TRUE, plot = FALSE)
          })
        }
        # Berechne den Fehler (gleason_r_1$value) für die aktuellen Startparameter
        current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

        # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
        if (!is.na(current_error)) {
          best_errors <- c(best_errors, current_error)
          best_starts <- rbind(best_starts, start_params)
        }
      }
    }
  }

  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}




find_best_start_4parameter <- function(p, q, max_shape, max_scale, max_rate,
                                       max_mix, steps_shape, steps_scale,
                                       steps_rate, steps_mix,fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
  best_starts <- matrix(nrow = 0, ncol = 4)  # Matrix für Startparameter

  for (shape in seq(0, max_shape, by = steps_shape)) {
    for (scale in seq(0, max_scale, by = steps_scale)) {
      for (rate in seq(0, max_rate, by = steps_rate)) {
```

```
      for (mix in seq(0, max_mix, by = steps_mix)) {
        start_params <- c(shape, scale, rate, mix)

        # Schätze die Weibull-Parameter mit den aktuellen Startparametern
        output <- capture.output({
          result <- fitting_function(p = p, q = q, start = start_params,
                                     show.output = TRUE, plot = FALSE)
        })

        # Berechne den Fehler (gleason_r_1$value) für die aktuellen Startparameter
        current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

        # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
        if (!is.na(current_error)) {
          best_errors <- c(best_errors, current_error)
          best_starts <- rbind(best_starts, start_params)
        }
      }
    }
  }
}

# Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
best_index <- which.min(best_errors)

# Wähle den besten Startparameter mit dem kleinsten Fehler aus
best_start <- best_starts[best_index, ]

# Gib den besten Startparameter und den entsprechenden Fehler aus
cat("Bester Startparameter:", best_start, "\n")
cat("Bester Fehler:", best_errors[best_index], "\n")

return(best_start)
}
```

Datenanpassung an die Daten von Gleason

Weibullverteilung

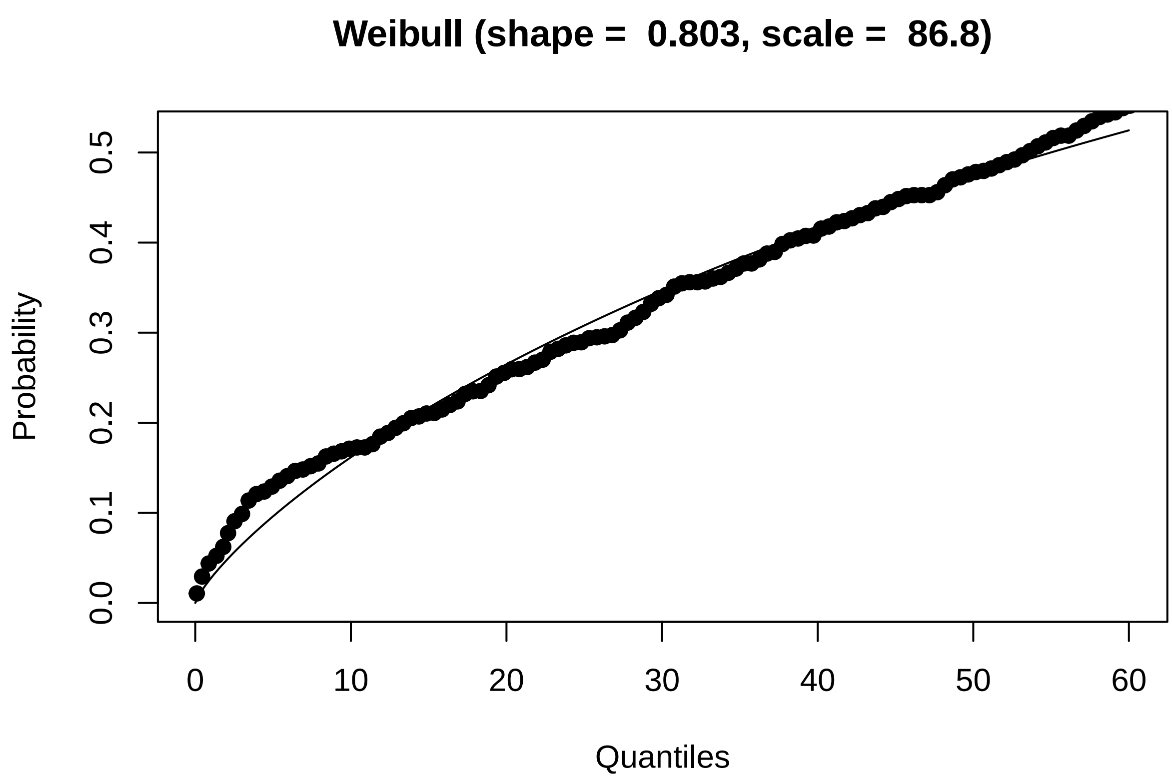
```
# Weibullverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getweibullpar.R")

best_gleason_r_1 <- find_best_start_2parameter(p = yr_gleason/100, q = xr_gleason,
                                               max_beta = 10, max_eta = 10,
                                               steps_beta = 1, steps_eta = 1,
                                               fitting_function = getweibullpar)
```

```
## Bester Startparameter: 0 0
## Bester Fehler: 1.903876e-06
```

```
gleason_r_1 <- getweibullpar(
                        p = yr_gleason/100,
                        q = xr_gleason,
```

```r
                        start = best_gleason_r_1,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1]  0.8030507 86.7753367
##
## $value
## [1] 1.903876e-06
##
## $counts
## function gradient
##       64       64
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

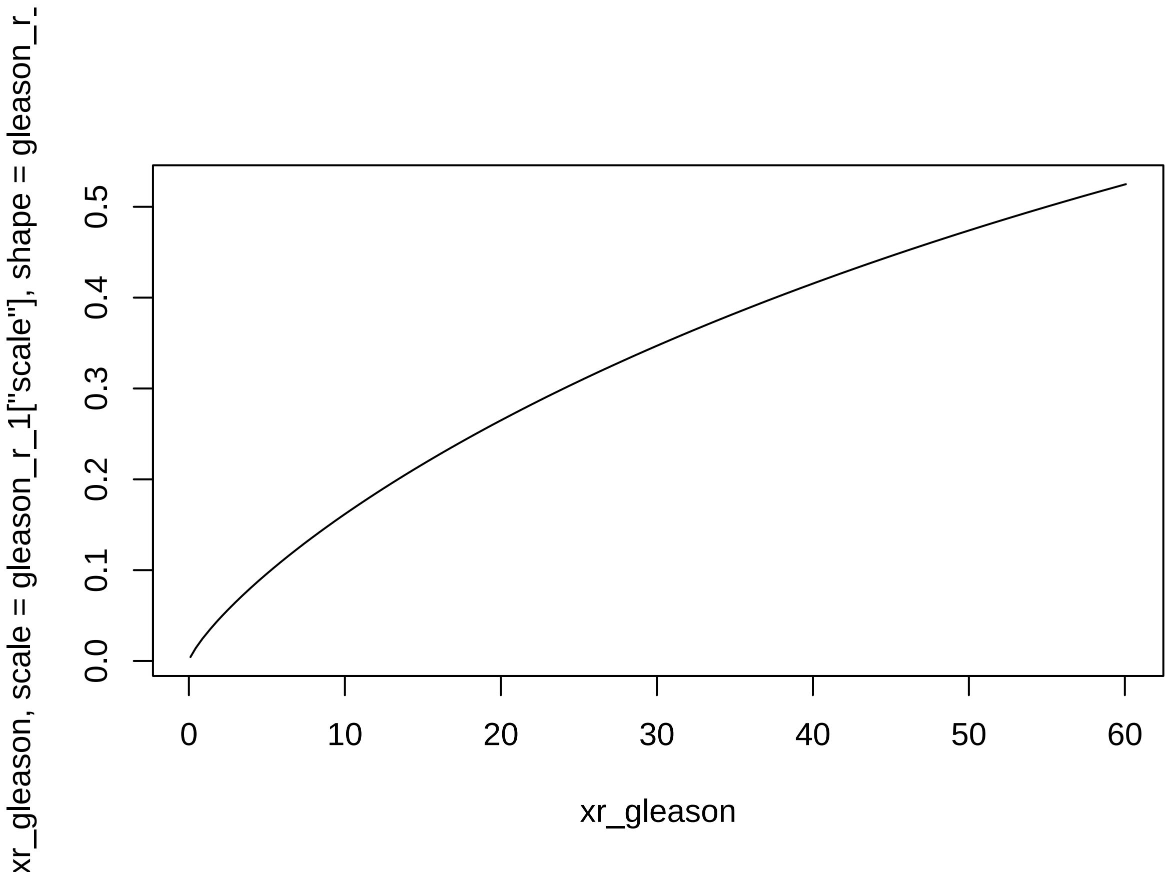


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```r
gleason_r_1
```

```
##     shape      scale
##  0.8030507 86.7753367
```

```r
plot(xr_gleason,
     pweibull(xr_gleason,
              scale = gleason_r_1["scale"],
              shape = gleason_r_1["shape"]), type = "l")
```



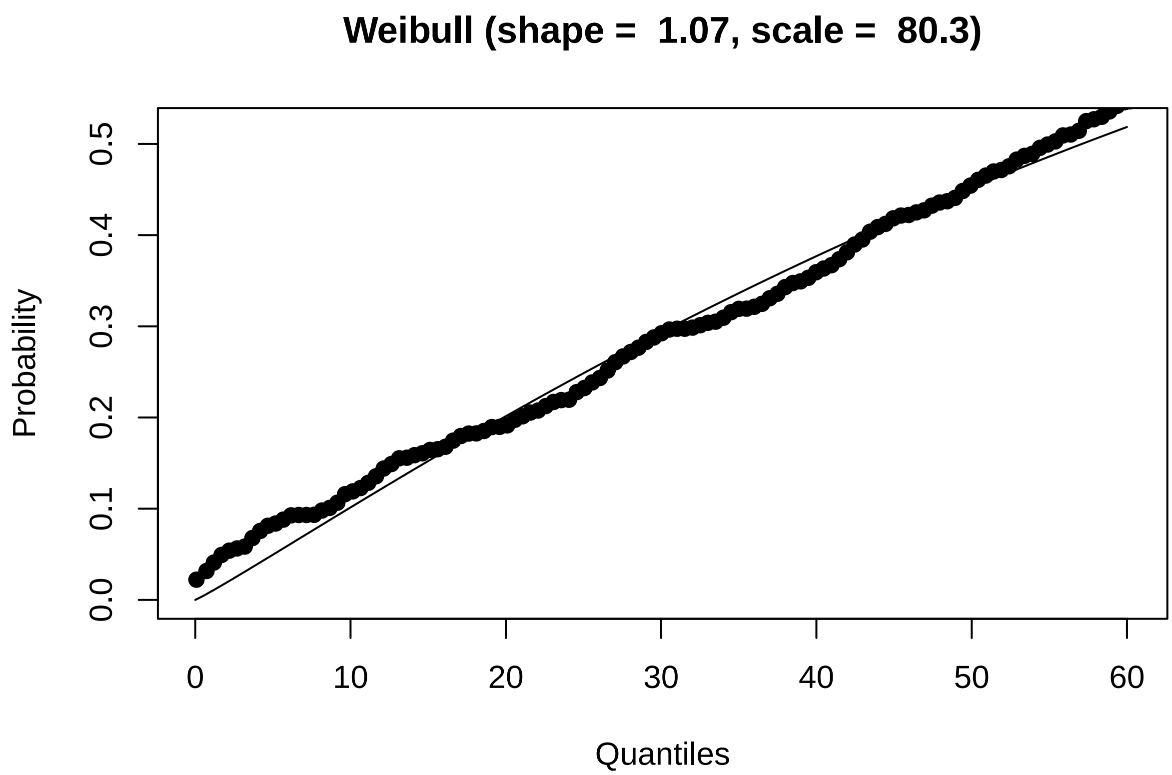

```r
best_gleason_b_1 <- find_best_start_2parameter(p = yb_gleason/100, q = xb_gleason,
                                               max_beta = 10, max_eta = 10,
                                               steps_beta = 1, steps_eta = 1,
                                               fitting_function = getweibullpar)
```

```
## Bester Startparameter: 1 3
## Bester Fehler: 2.32376e-06
```

```r
gleason_b_1 <- getweibullpar(
                        p = yb_gleason/100,
                        q = xb_gleason,
                        start = best_gleason_b_1,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

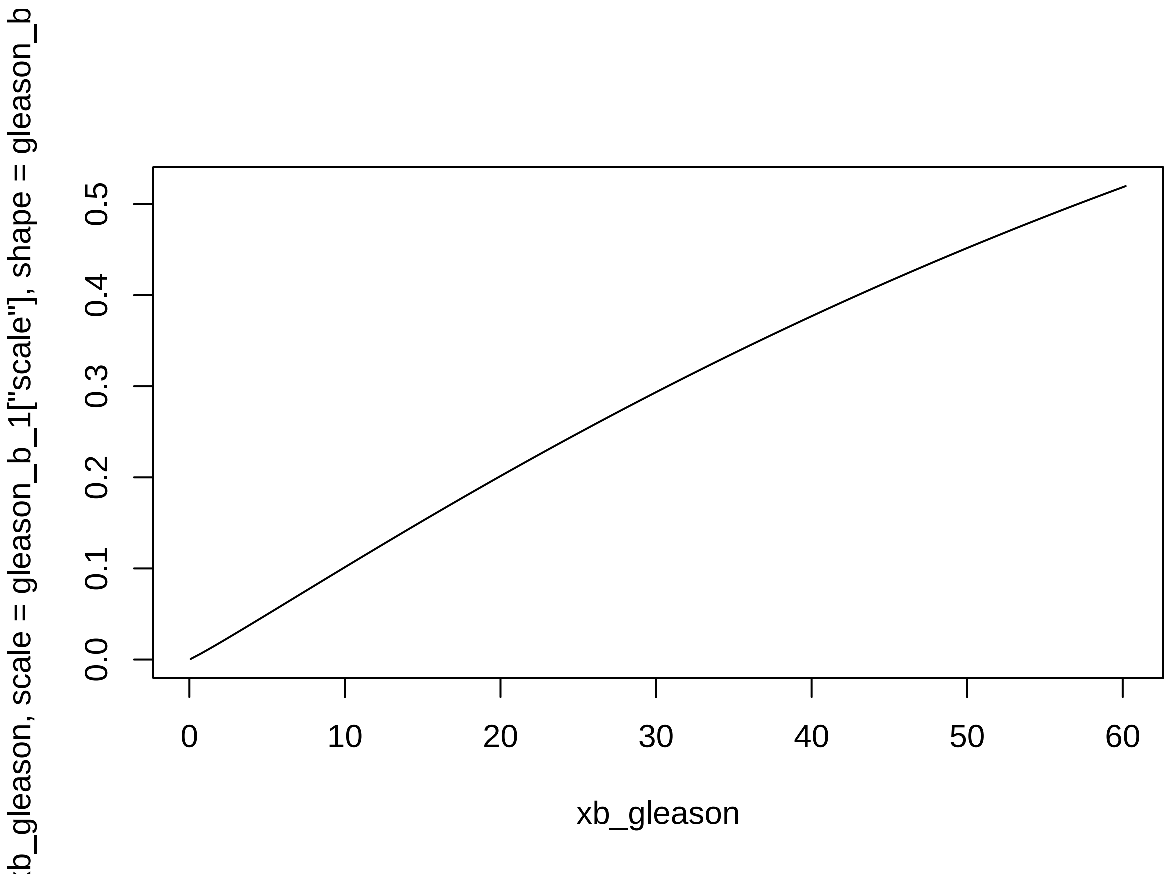
```
## $par
## [1]  1.07295 80.34824
##
## $value
## [1] 2.32376e-06
##
```

```
## $counts
## function gradient
##       77       77
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```r
plot(xb_gleason,
     pweibull(xb_gleason,
              scale = gleason_b_1["scale"],
              shape = gleason_b_1["shape"]), type = "l")
```



Exponentiierte Weibullverteilung

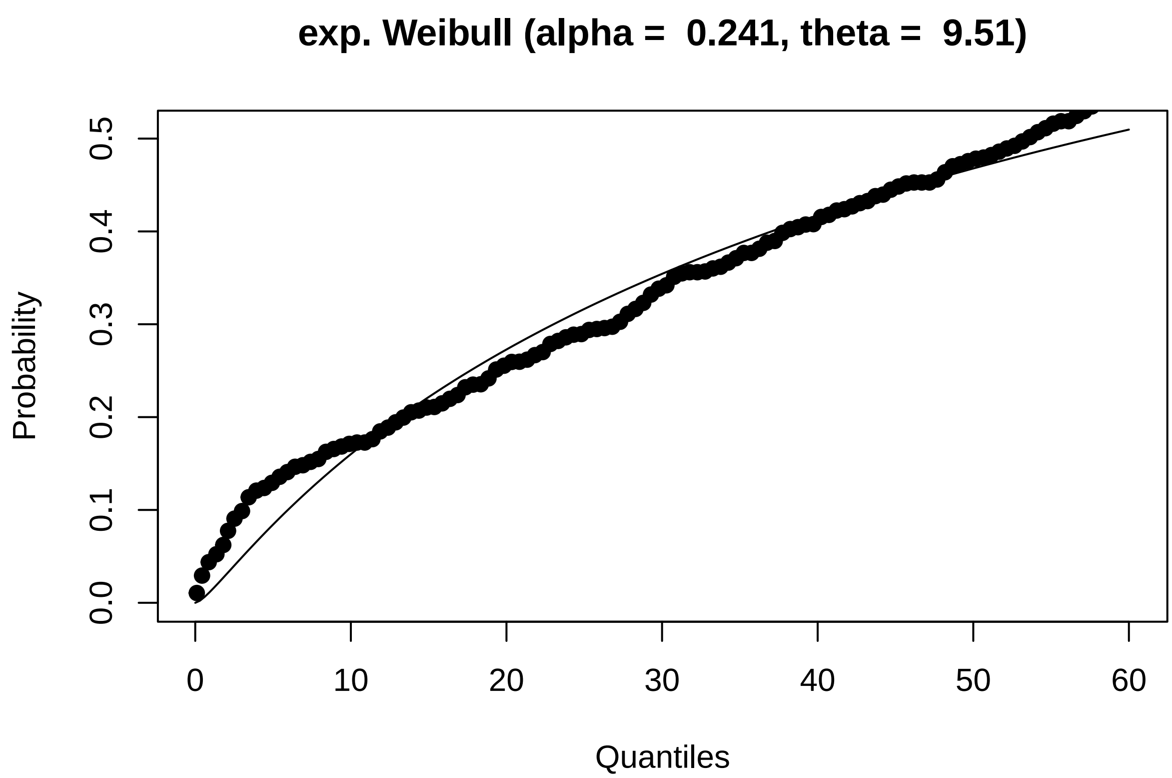
```r
# exponentiierte Weibullverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getweibpar.R")

best_weibbull_xr_gleason <- find_best_start_2parameter(p = yr_gleason/100,
                                                       q = xr_gleason,
                                                       max_beta = 10,
                                                       max_eta = 10,
                                                       steps_beta = 1,
                                                       steps_eta = 1,
                                                       fitting_function = getweibpar)
```

```
## Bester Startparameter: 2 3
## Bester Fehler: 4.240118e-06
```

```r
weibbull_xr_gleason <- getweibpar(
                        p = yr_gleason/100,
                        q = xr_gleason,
                        start = best_weibbull_xr_gleason,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.2409727 9.5108123
```

```
##
## $value
## [1] 4.240118e-06
##
## $counts
## function gradient
##       31       31
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

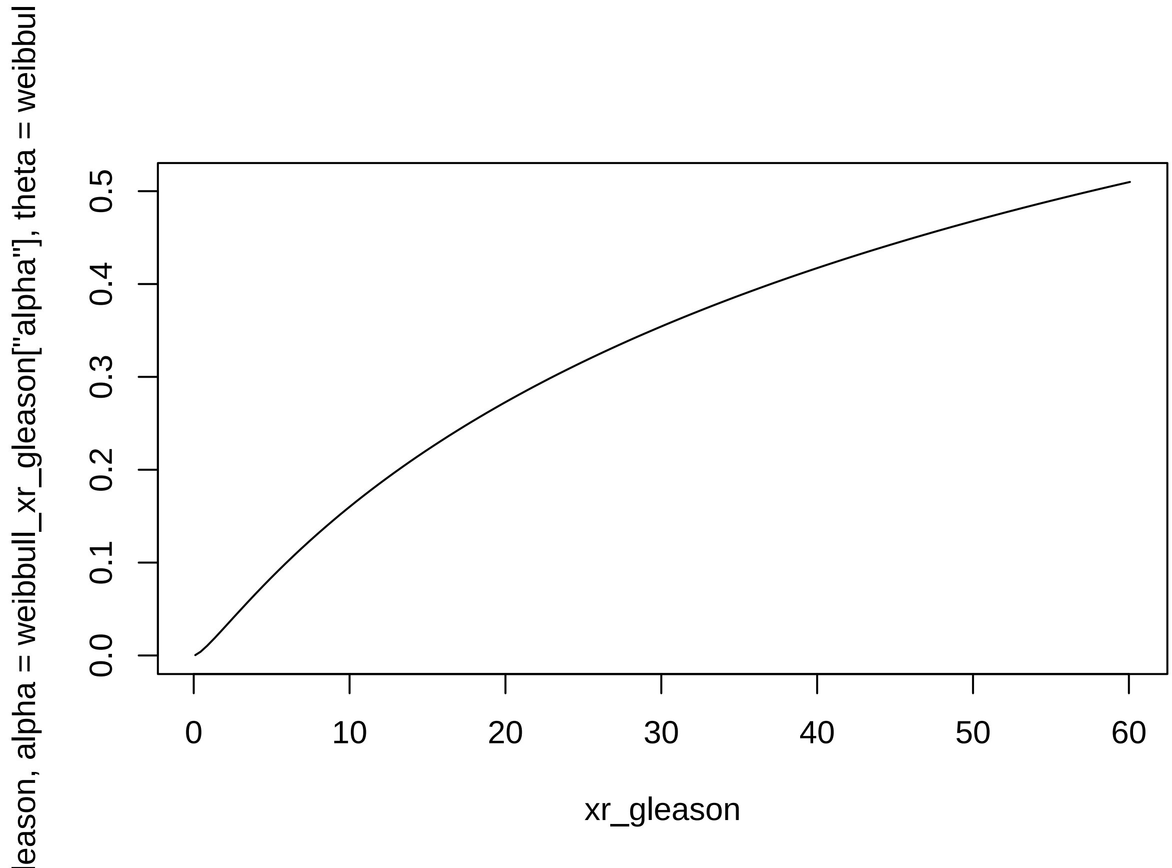


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
weibull_xr_gleason
```

```
##     alpha     theta
## 0.2409727 9.5108123
```

```
plot(xr_gleason,
     pexpo.weibull(xr_gleason,
              alpha = weibull_xr_gleason ["alpha"],
              theta = weibull_xr_gleason ["theta"]), type = "l")
```



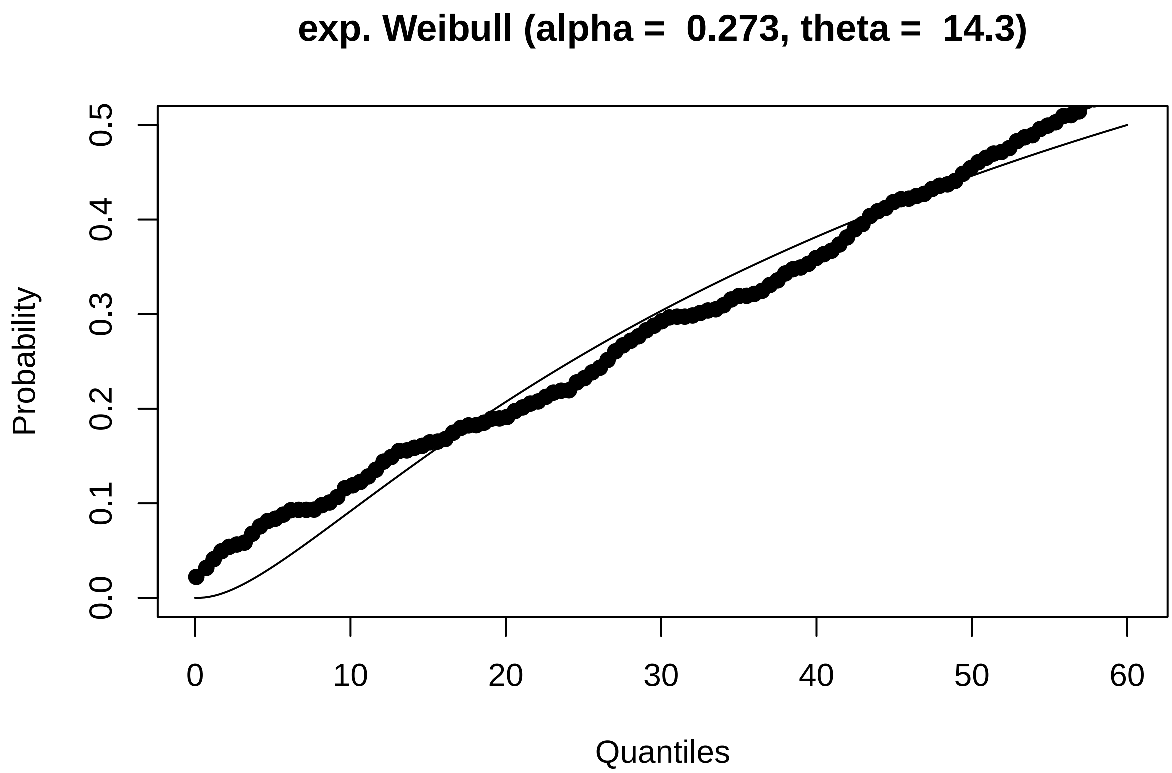

```
best_weibbull_xb_gleason <- find_best_start_2parameter(p = yb_gleason/100,
                                                       q = xb_gleason,
                                                       max_beta = 10,
                                                       max_eta = 10,
                                                       steps_beta = 1,
                                                       steps_eta = 1,
                                                       fitting_function = getweibpar)
```

```
## Bester Startparameter: 2 4
## Bester Fehler: 5.363296e-06
```

```
weibbull_xb_gleason  <- getweibpar(
                          p = yb_gleason/100,
                          q = xb_gleason,
                          start = best_weibbull_xb_gleason,
                          show.output = TRUE,
                          plot = TRUE
                        )
```

```
## $par
## [1]  0.272552 14.322883
##
## $value
## [1] 5.363296e-06
##
## $counts
## function gradient
```

```
##         32        32
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

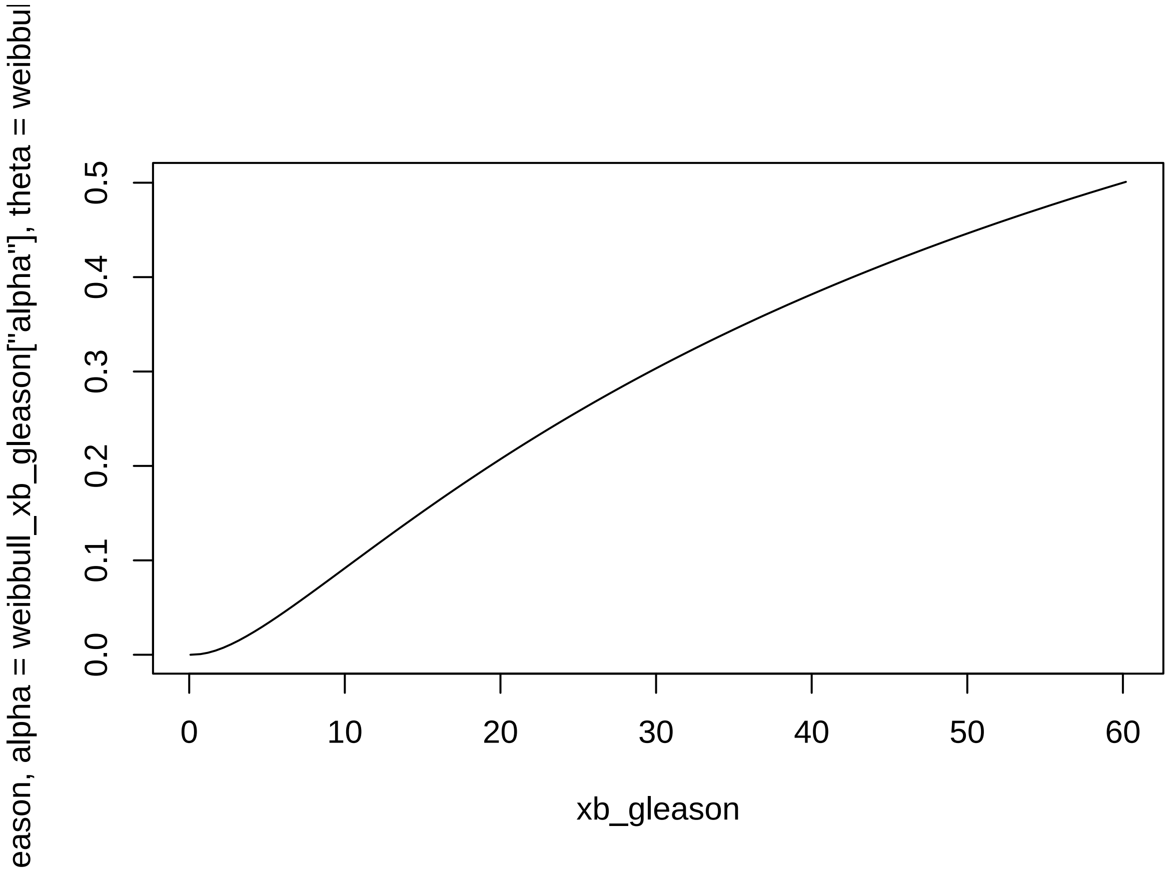


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```
weibbull_xb_gleason
```

```
##      alpha     theta
##  0.272552 14.322883
```

```
plot(xb_gleason,
     pexpo.weibull(xb_gleason,
              alpha = weibbull_xb_gleason ["alpha"],
              theta = weibbull_xb_gleason ["theta"]), type = "l")
```



Mischung aus Weibull- und Exponentialverteilung

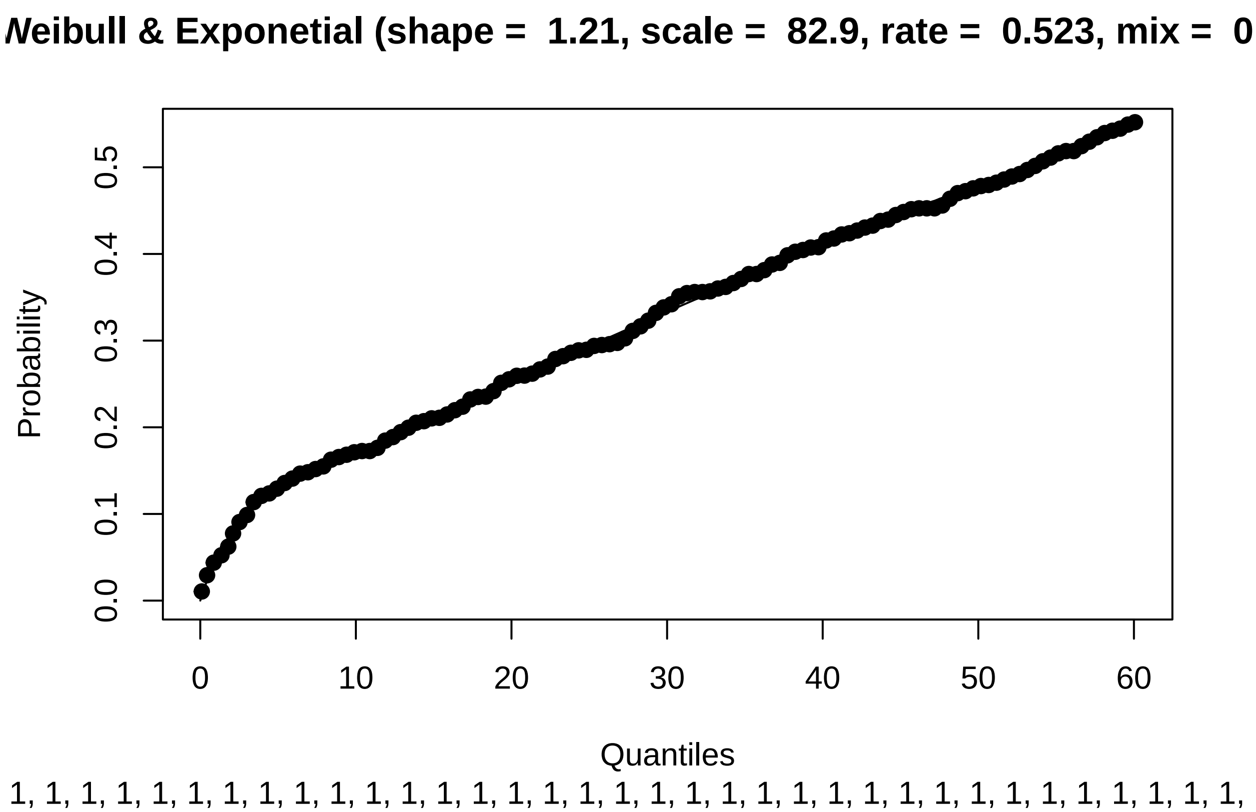
```
# Mischung aus Weibull- und Exponentialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getweibex.R")

best_weibex_xr_gleason <- find_best_start_4parameter(p = yr_gleason/100,
                                                     q = xr_gleason,
                                                     max_shape = 1,
                                                     max_scale = 100,
                                                     max_rate = 0.7,
                                                     max_mix = 1,
                                                     steps_shape = 0.1,
                                                     steps_scale = 20,
                                                     steps_rate = 0.1,
                                                     steps_mix = 0.1,
                                                     fitting_function = getweibex)
```

```
## Bester Startparameter: 0.5 60 0.4 0.1
## Bester Fehler: 1.537623e-07
```

```
weibex_xr_gleason <- getweibex(
                        p = yr_gleason/100,
                        q = xr_gleason,
                        start = best_weibex_xr_gleason, # c(0.5, 60, 0.4, 0.1),
                        show.output = TRUE,
                        plot = TRUE
```

```
                      )
```

```
## $par
## [1]  1.2051344 82.8901131  0.5230777  2.1375789
##
## $value
## [1] 1.537623e-07
##
## $counts
## function gradient
##       48       48
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

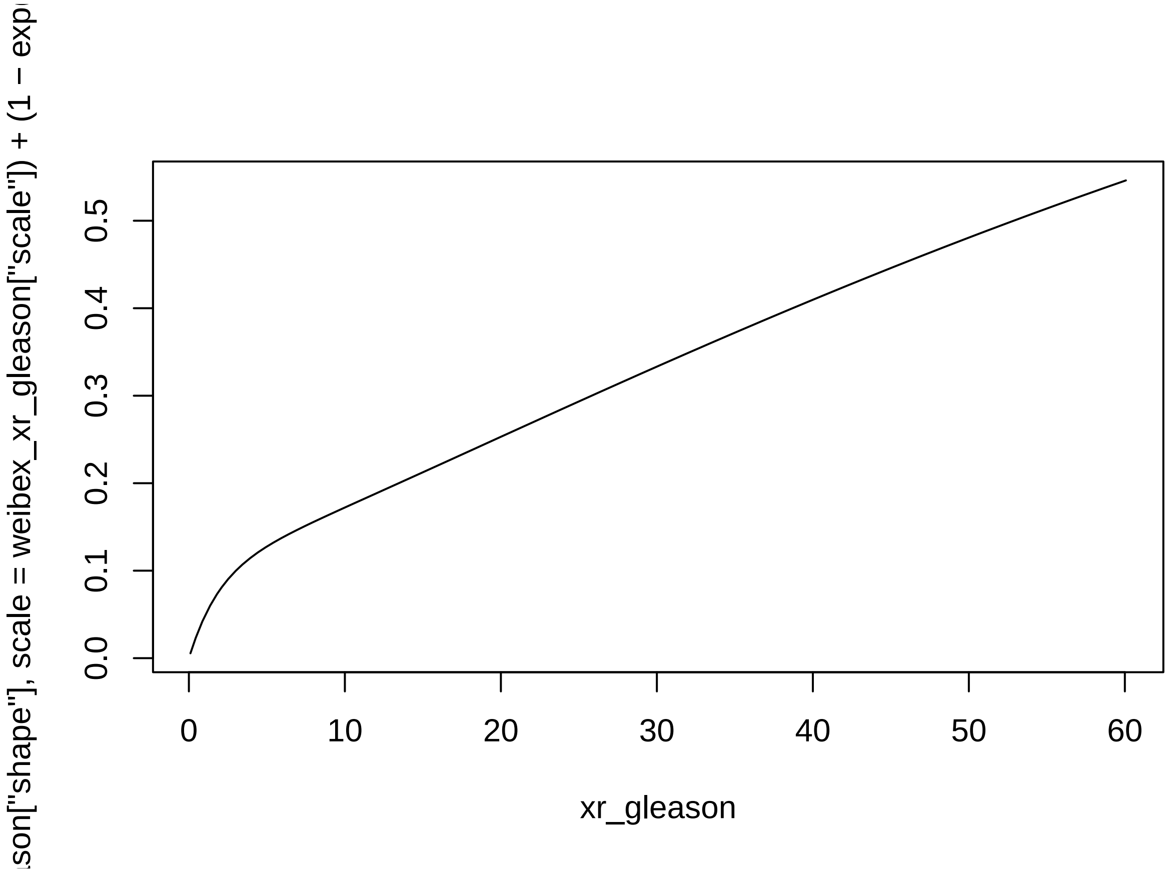



```
weibex_xr_gleason
```

```
##      shape      scale       rate        mix
##  1.2051344 82.8901131  0.5230777  2.1375789
```

```
# Wenn Ergebnisse aus weibex_xr1_1 von Funktion abgelesen
plot(xr_gleason,
     (exp(weibex_xr_gleason["mix"]) / ( 1 + exp(weibex_xr_gleason["mix"])) *
        stats::pweibull(q = xr_gleason,
```

```
                         shape = weibex_xr_gleason["shape"],
                         scale = weibex_xr_gleason["scale"]) +
        (1 - exp(weibex_xr_gleason["mix"]) / ( 1 + exp(weibex_xr_gleason["mix"]))) *
        stats::pexp(q = xr_gleason,
                     rate = weibex_xr_gleason["rate"])),
     type = "l")
```



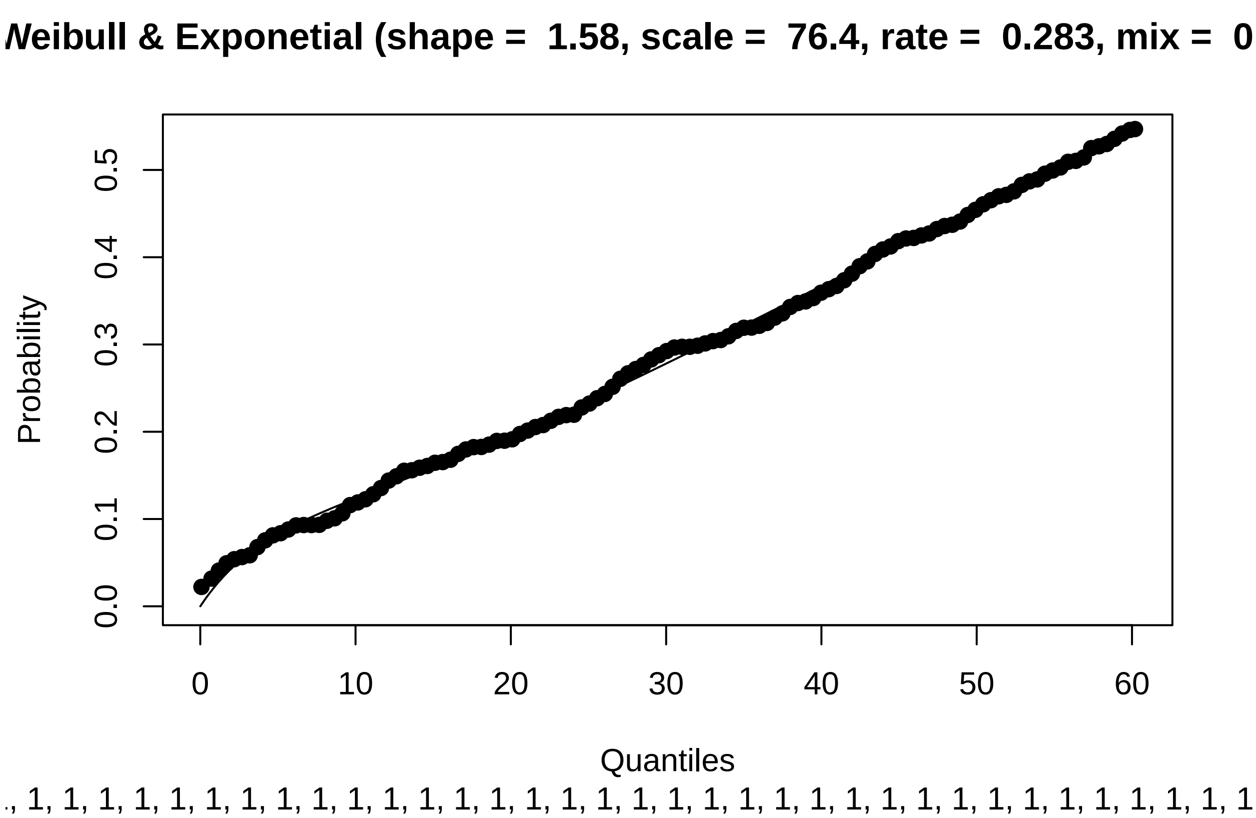

```
best_weibex_xb_gleason <- find_best_start_4parameter(p = yb_gleason/100,
                                                     q = xb_gleason,
                                                     max_shape = 1,
                                                     max_scale = 100,
                                                     max_rate = 0.7,
                                                     max_mix = 1,
                                                     steps_shape = 0.1,
                                                     steps_scale = 20,
                                                     steps_rate = 0.1,
                                                     steps_mix = 0.1,
                                                     fitting_function = getweibex)
```

```
## Bester Startparameter: 0.5 60 0.7 0.1
## Bester Fehler: 3.401223e-07
```

```
weibex_xb_gleason <- getweibex(
                         p = yb_gleason/100,
                         q = xb_gleason,
                         start = best_weibex_xb_gleason, # c(0.5, 60, 0.7, 0.1),
                         show.output = TRUE,
```

```
                        plot = TRUE
                      )
```

```
## $par
## [1]  1.5831633 76.3685496  0.2831678  2.2726626
##
## $value
## [1] 3.401223e-07
##
## $counts
## function gradient
##       29       29
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

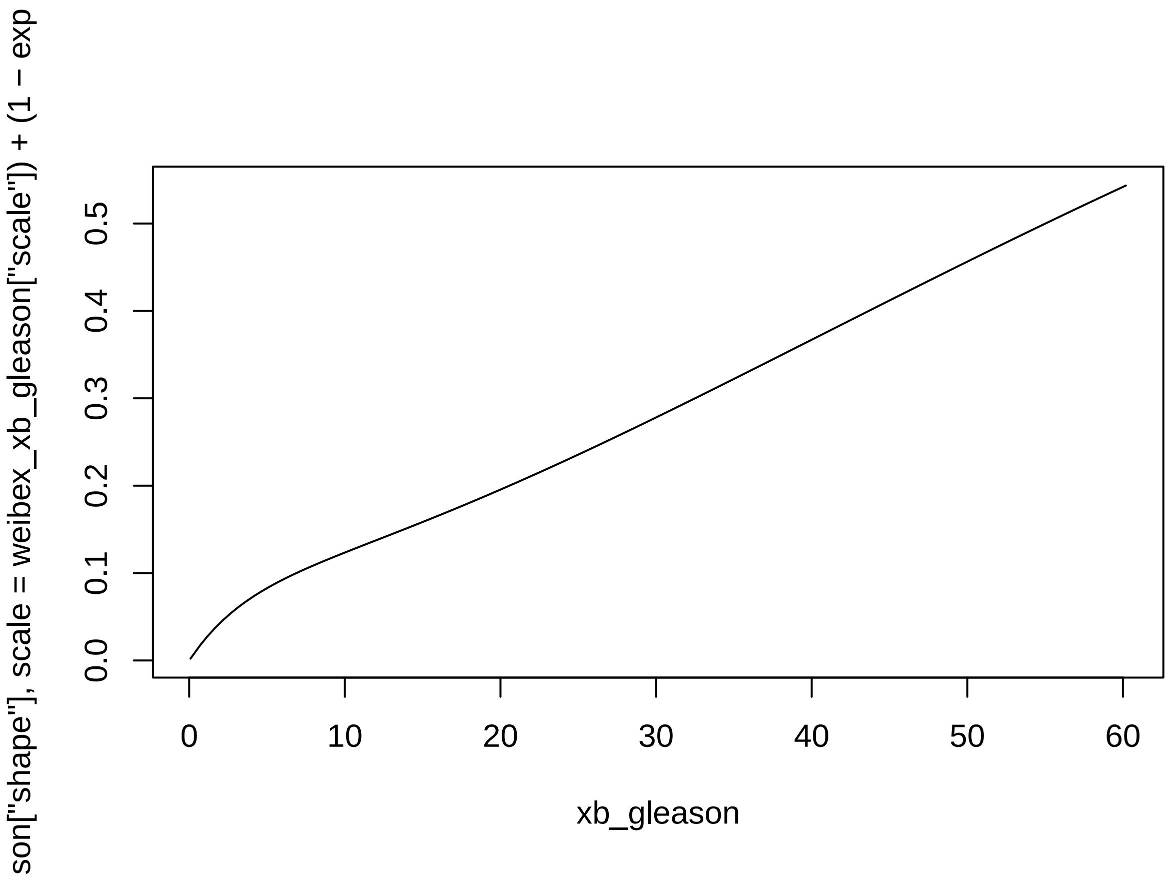



```
weibex_xb_gleason
```

```
##      shape      scale       rate        mix
##  1.5831633 76.3685496  0.2831678  2.2726626
```

```
plot(xb_gleason,
     (exp(weibex_xb_gleason["mix"]) / ( 1 + exp(weibex_xb_gleason["mix"])) *
        stats::pweibull(q = xb_gleason,
```

```r
                        shape = weibex_xb_gleason["shape"],
                        scale = weibex_xb_gleason["scale"]) +
        (1 - exp(weibex_xb_gleason["mix"]) / ( 1 + exp(weibex_xb_gleason["mix"]))) *
        stats::pexp(q = xb_gleason,
                    rate = weibex_xb_gleason["rate"])),
     type = "l")
```



Mischung von exponentiierter Weibull- und Exponentialverteilung

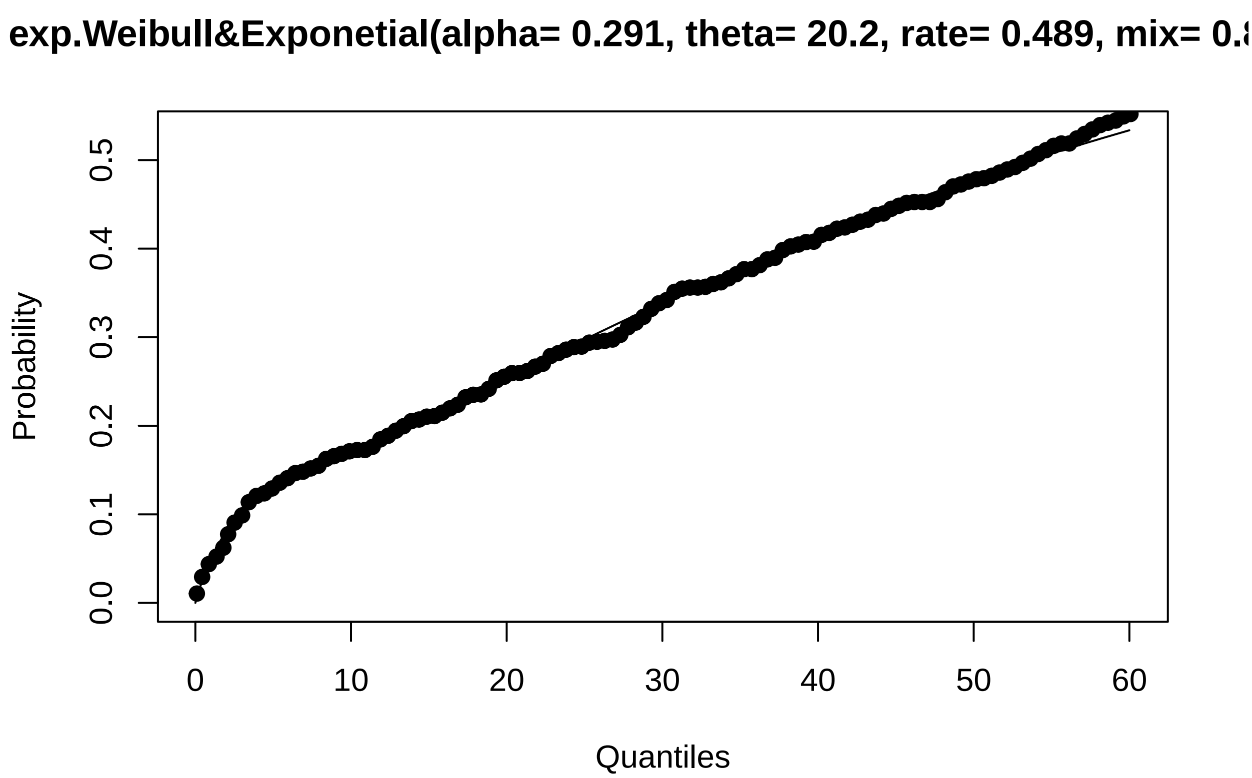
```r
# Mischung von exponentiierter Weibull- und Exponentialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/get2weibex.R")

best_weibex2_xr_gleason <- find_best_start_4parameter(p = yr_gleason/100,
                                                      q = xr_gleason,
                                                      max_shape = 1,
                                                      max_scale = 100,
                                                      max_rate = 0.7,
                                                      max_mix = 1,
                                                      steps_shape = 0.1,
                                                      steps_scale = 20,
                                                      steps_rate = 0.1,
                                                      steps_mix = 0.1,
                                                      fitting_function = get2weibex)
```

```
## Bester Startparameter: 0.2 20 0.5 0.3
```

```
## Bester Fehler: 3.069899e-07
```

```r
weibex2_xr_gleason <- get2weibex(
                        p = yr_gleason/100,
                        q = xr_gleason,
                        start = best_weibex2_xr_gleason,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1]  0.2910099 20.2104315  0.4890507  1.9162629
##
## $value
## [1] 3.069899e-07
##
## $counts
## function gradient
##       47       47
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

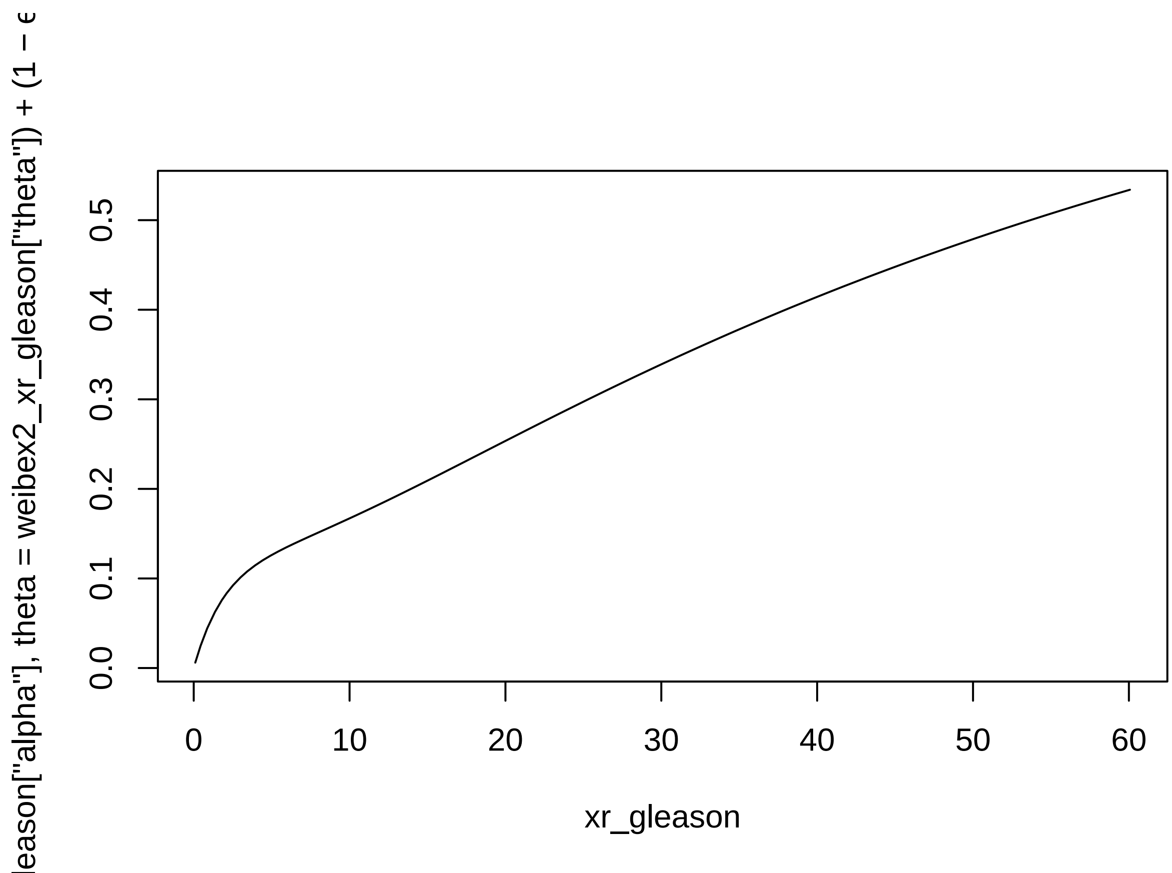


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```r
weibex2_xr_gleason
```

```
##       alpha      theta       rate        mix
##  0.2910099 20.2104315  0.4890507  1.9162629
```

```r
plot(xr_gleason,
     (exp(weibex2_xr_gleason["mix"]) / ( 1 + exp(weibex2_xr_gleason["mix"])) *
        reliaR::pexpo.weibull(q = xr_gleason,
                              alpha = weibex2_xr_gleason["alpha"],
                              theta = weibex2_xr_gleason["theta"]) +
        (1 - exp(weibex2_xr_gleason["mix"]) / ( 1 + exp(weibex2_xr_gleason["mix"]))) *
        stats::pexp(q = xr_gleason,
                    rate = weibex2_xr_gleason["rate"])),
     type = "l")
```

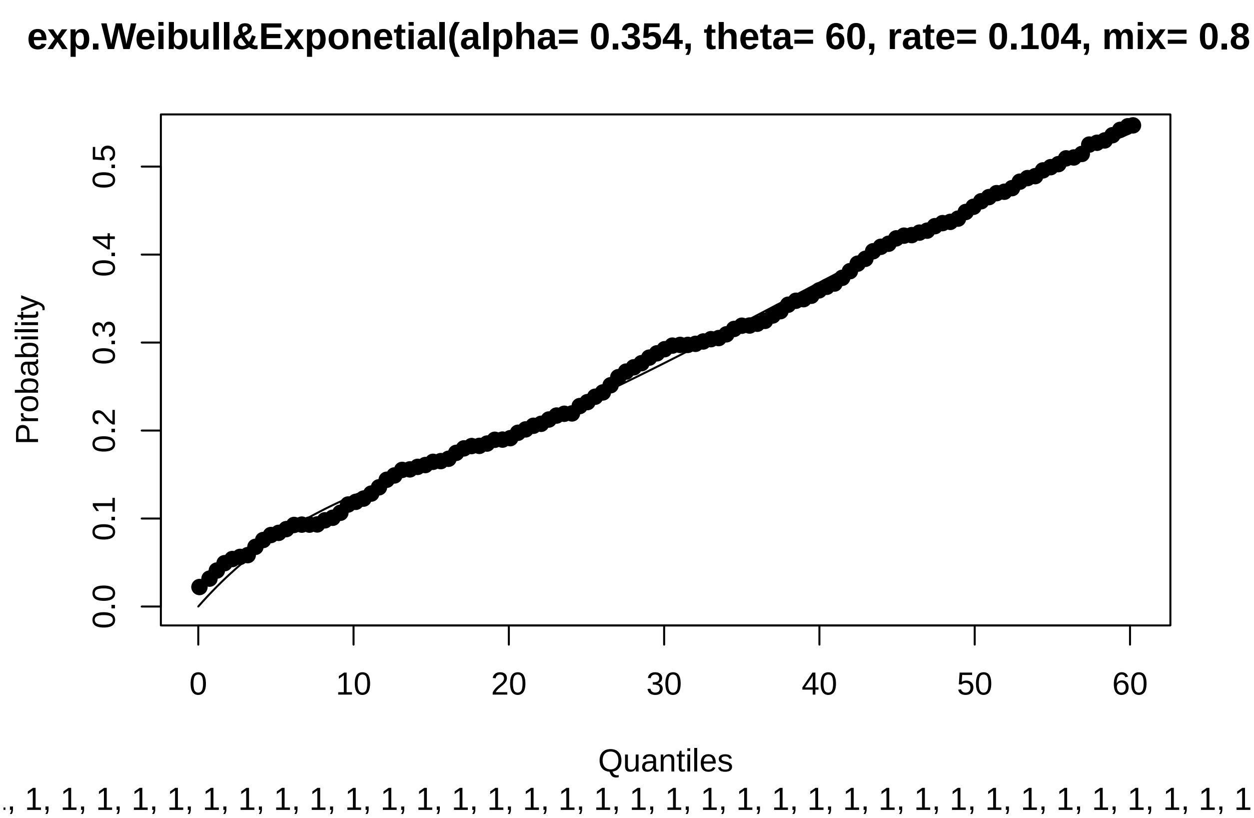



```r
best_weibex2_xb_gleason <- find_best_start_4parameter(p = yb_gleason/100,
                                                      q = xb_gleason,
                                                      max_shape = 1,
                                                      max_scale = 100,
                                                      max_rate = 0.7,
                                                      max_mix = 1,
                                                      steps_shape = 0.1,
                                                      steps_scale = 20,
                                                      steps_rate = 0.1,
                                                      steps_mix = 0.1,
                                                      fitting_function = get2weibex)
```

```
## Bester Startparameter: 0.2 60 0.6 0.3
## Bester Fehler: 4.475831e-07
```

```
weibex2_xb_gleason <- get2weibex(
                        p = yb_gleason/100,
                        q = xb_gleason,
                        start = best_weibex2_xb_gleason,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

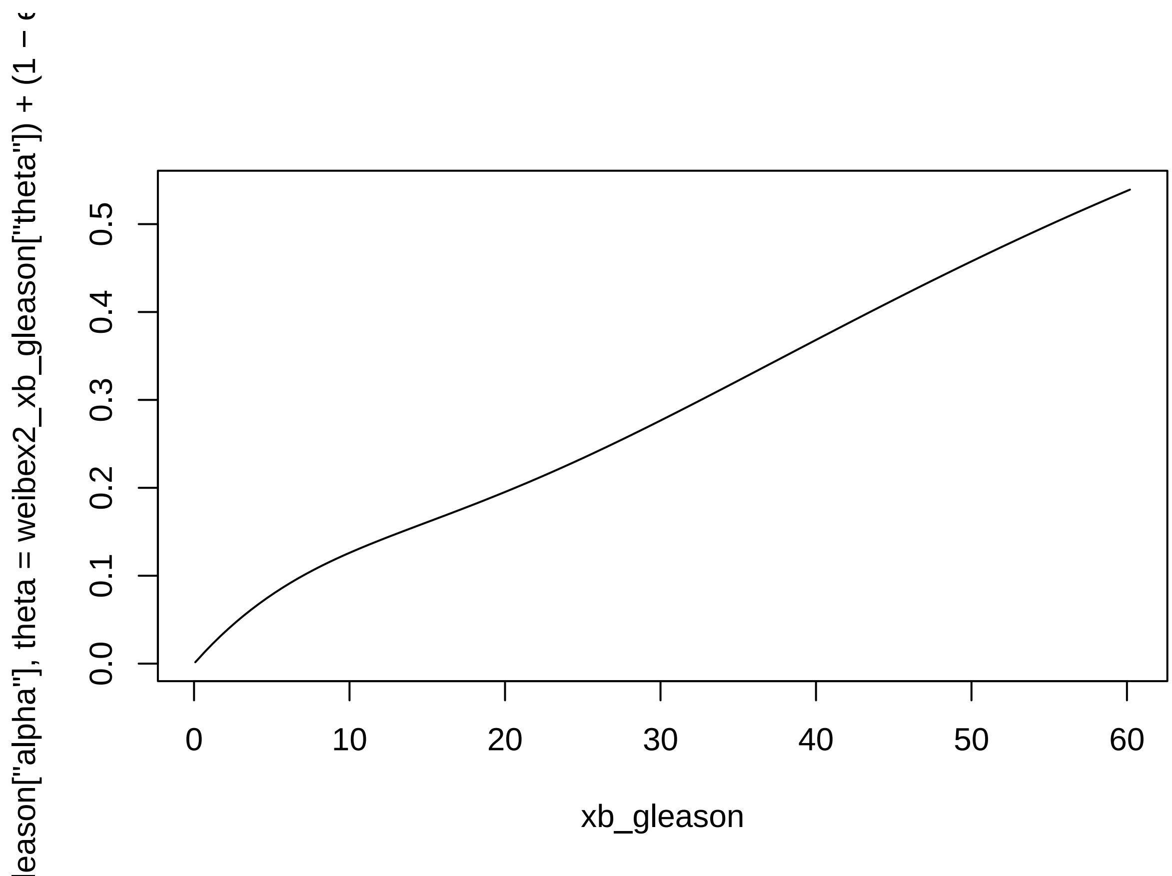
```
## $par
## [1]  0.3541876 59.9990336  0.1038148  1.4276088
##
## $value
## [1] 4.475831e-07
##
## $counts
## function gradient
##       23       23
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



```
weibex2_xb_gleason
```

```
##      alpha      theta       rate        mix
##  0.3541876 59.9990336  0.1038148  1.4276088
```

```
plot(xb_gleason,
     (exp(weibex2_xb_gleason["mix"]) / ( 1 + exp(weibex2_xb_gleason["mix"])) *
        reliaR::pexpo.weibull(q = xb_gleason,
                              alpha = weibex2_xb_gleason["alpha"],
                              theta = weibex2_xb_gleason["theta"]) +
        (1 - exp(weibex2_xb_gleason["mix"]) / ( 1 + exp(weibex2_xb_gleason["mix"]))) *
        stats::pexp(q = xb_gleason,
                    rate = weibex2_xb_gleason["rate"])),
     type = "l")
```



Exponentiierte Weibullverteilung ohne Lambda = 1

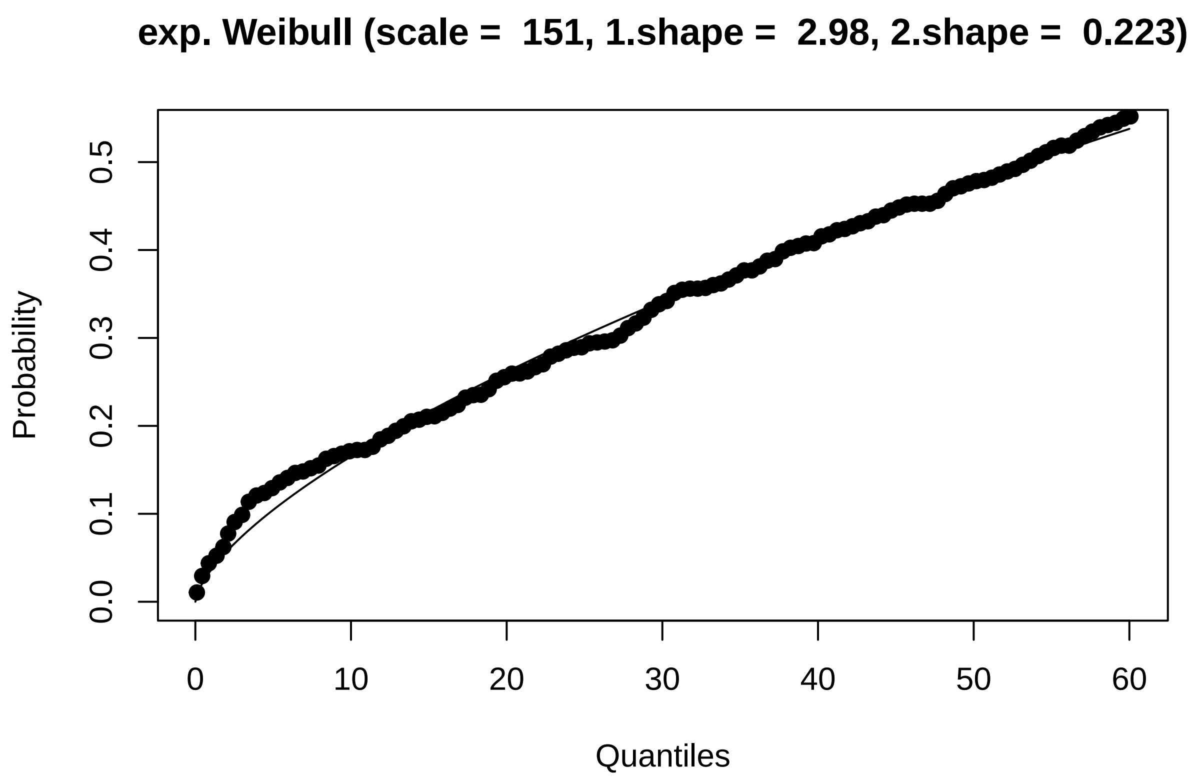
```
# exponentiierte Weibullverteilung ohne Lambda = 1
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getexpweib.R")

best_expweib_xr_gleason <- find_best_start_3parameter(p = yr_gleason/100,
                                                      q = xr_gleason,
                                                      max_shape1 = 10,
                                                      max_shape2 = 10,
                                                      max_scale = 10,
                                                      steps_shape1 = 1,
```

```r
                                                    steps_shape2 = 1,
                                                    steps_scale = 1,
                                                    fitting_function = getexpweib)
```

```
## Bester Startparameter: 7 9 7
## Bester Fehler: 9.048032e-07
```

```r
expweib_xr_gleason <- getexpweib(
                        p = yr_gleason/100,
                        q = xr_gleason,
                        start = best_expweib_xr_gleason,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 151.4219338   2.9757579   0.2226113
##
## $value
## [1] 9.048032e-07
##
## $counts
## function gradient
##       71       71
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

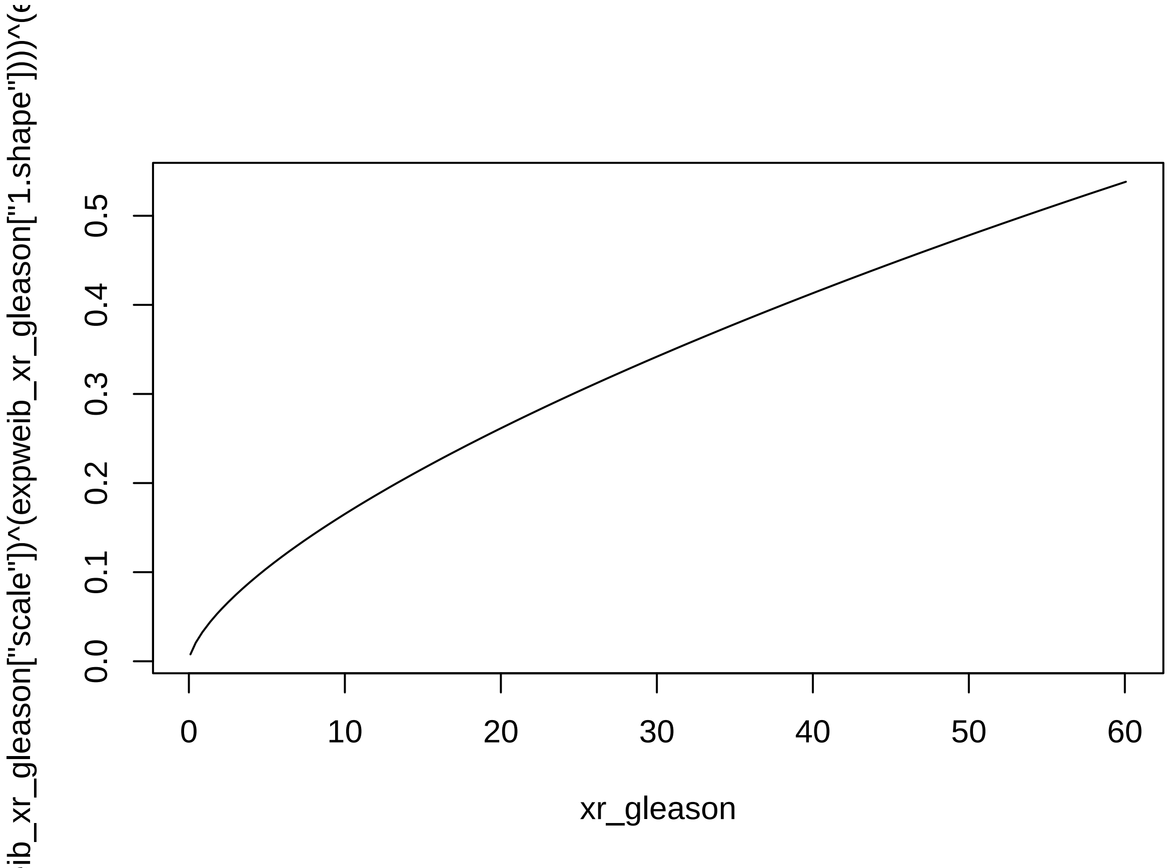


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
expweib_xr_gleason
```

```
##       scale     1.shape     2.shape
## 151.4219338   2.9757579   0.2226113
```

```
plot(xr_gleason,
     (1 - exp(-(xr_gleason / expweib_xr_gleason["scale"])^
                     (expweib_xr_gleason["1.shape"]))) ^ (expweib_xr_gleason["2.shape"]),
     type = "l")
```



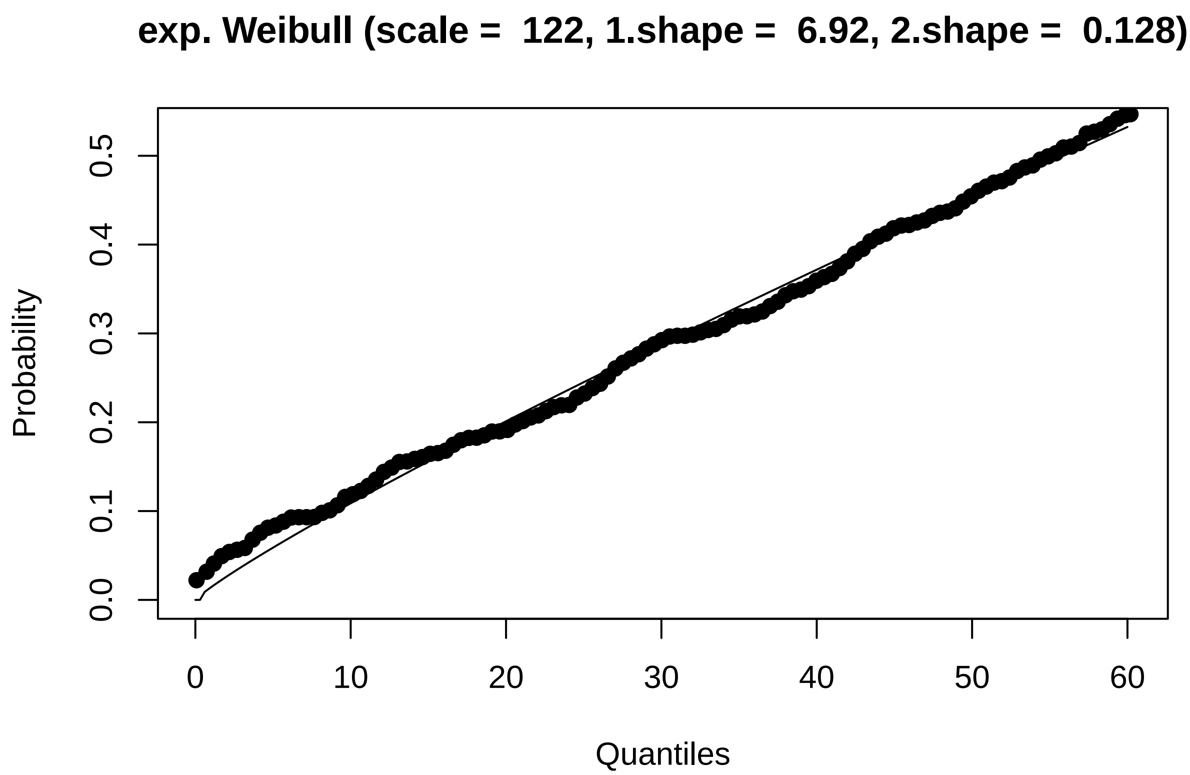

```
best_expweib_xb_gleason <- find_best_start_3parameter(p = yb_gleason/100,
                                                     q = xb_gleason,
                                                     max_shape1 = 10,
                                                     max_shape2 = 10,
                                                     max_scale = 10,
                                                     steps_shape1 = 1,
                                                     steps_shape2 = 1,
                                                     steps_scale = 1,
                                                     fitting_function = getexpweib)
```

```
## Bester Startparameter: 7 10 4
## Bester Fehler: 1.122354e-06
```

```
expweib_xb_gleason <- getexpweib(
                        p = yb_gleason/100,
                        q = xb_gleason,
                        start = best_expweib_xb_gleason,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 122.0821385   6.9216314   0.1281085
##
## $value
## [1] 1.122354e-06
##
```

```
## $counts
## function gradient
##       78       78
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

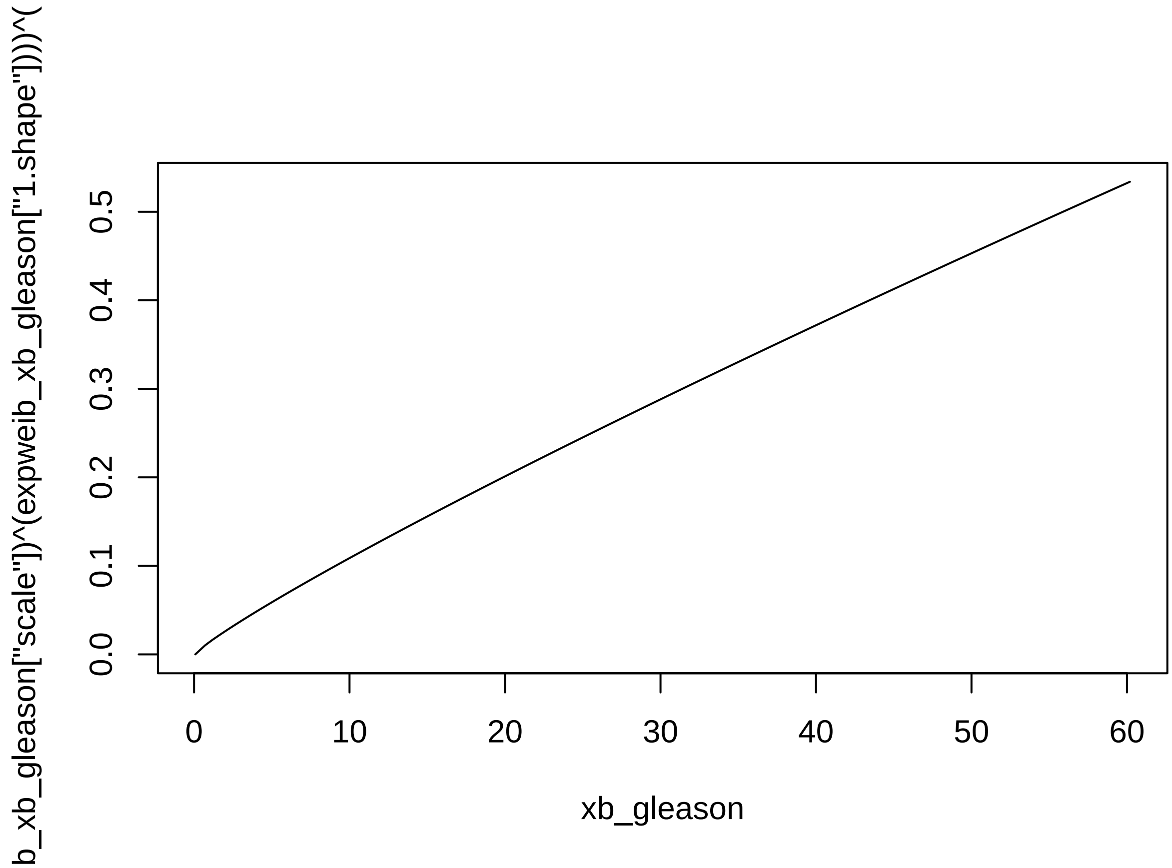


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```r
expweib_xb_gleason
```

```
##       scale     1.shape     2.shape
## 122.0821385   6.9216314   0.1281085
```

```r
plot(xb_gleason,
     (1 - exp(-(xb_gleason / expweib_xb_gleason["scale"])^(expweib_xb_gleason["1.shape"])))^(expweib_xb_
     type = "l")
```



3-Stufige Exponetialverteilung

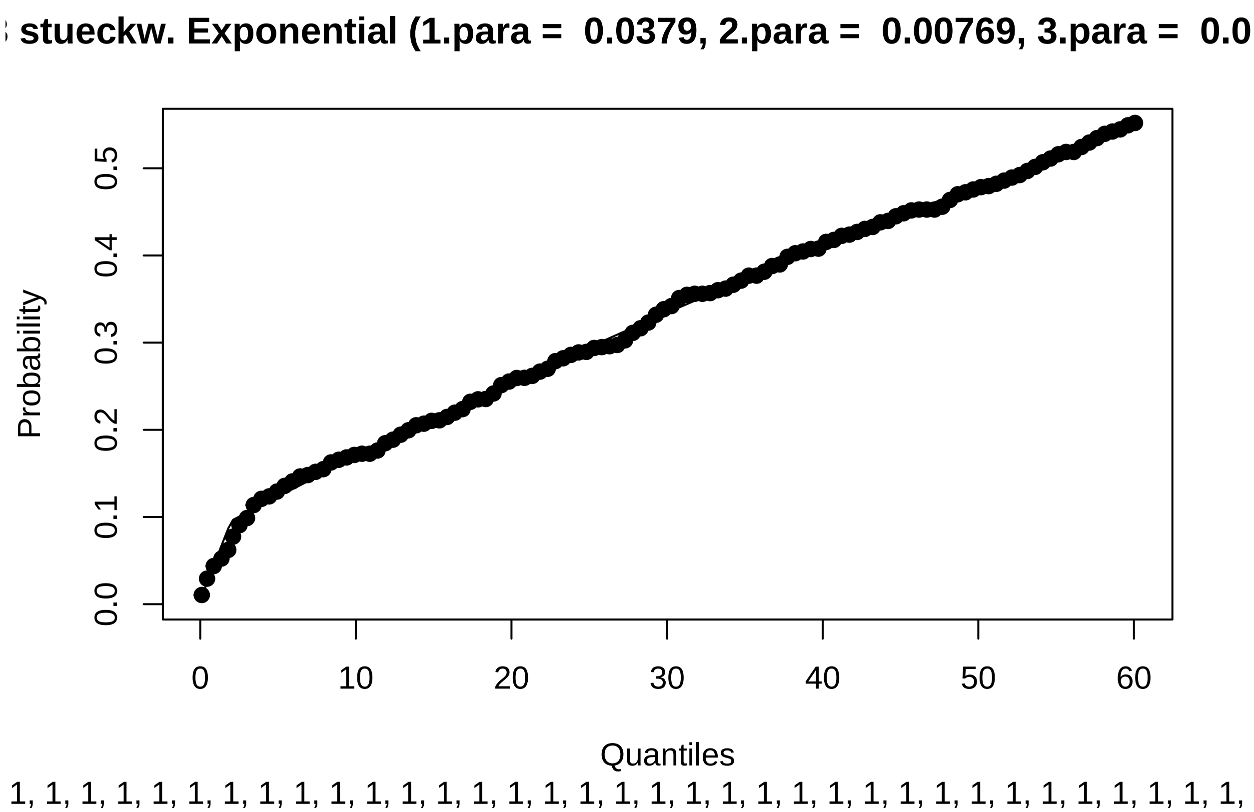
```r
# 3-Stufige Exponetialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getstuexp3.R")

best_stuexp3_xr_gleason <- find_best_start_3parameter(p = yr_gleason/100,
                                                    q = xr_gleason,
                                                    max_shape1 = 0.1,
                                                    max_shape2 = 0.1,
                                                    max_scale = 0.1,
                                                    steps_shape1 = 0.01,
                                                    steps_shape2 = 0.01,
                                                    steps_scale = 0.01,
                                                    fitting_function = getstuexp3)
```

```
## Bester Startparameter: 0.04 0.01 0.01
## Bester Fehler: 2.502095e-07
```

```r
stuexp3_xr_gleason <- getstuexp3(
                        p = yr_gleason/100,
                        q = xr_gleason,
                        start = best_stuexp3_xr_gleason,
                        show.output = TRUE,
                        plot = TRUE,
                        wert1 = 2,
                        wert2 = 6
```

```
                      )
```

```
## $par
## [1] 0.037861420 0.007691976 0.001752558
##
## $value
## [1] 2.502095e-07
##
## $counts
## function gradient
##       32       32
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

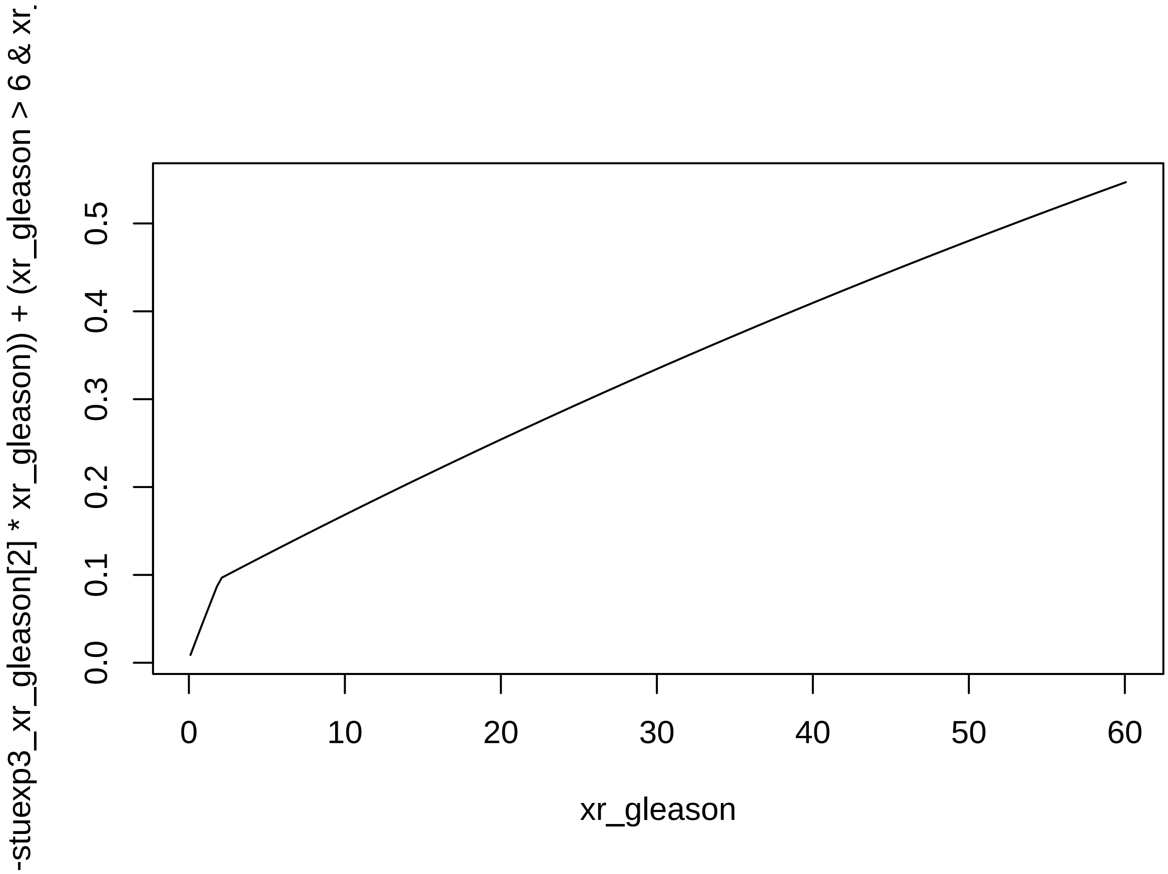



```
stuexp3_xr_gleason
```

```
##      1.para      2.para      3.para
## 0.037861420 0.007691976 0.001752558
```

```
plot(xr_gleason,
     ((xr_gleason > 0 & xr_gleason <= 2 ) * (1 - exp(-stuexp3_xr_gleason[1] * xr_gleason)) +
                              (xr_gleason > 2 & xr_gleason <= 6 ) *
             (1 - exp(-stuexp3_xr_gleason[1] * 2))  +
```

```
                                    (exp(-2 * stuexp3_xr_gleason[2]) -
                                             exp(-stuexp3_xr_gleason[2] * xr_gleason)) +
                                    (xr_gleason > 6 & xr_gleason <= 65 ) *
               (1 - exp(-stuexp3_xr_gleason[1] * 2)) +
                                    (exp(-2 * stuexp3_xr_gleason[2]) - exp(-6 * stuexp3_xr_gleason[2])) +
                                    (exp(-6 * stuexp3_xr_gleason[3]) -
                                             exp(-stuexp3_xr_gleason[3] * xr_gleason))),
     type = "l")
```



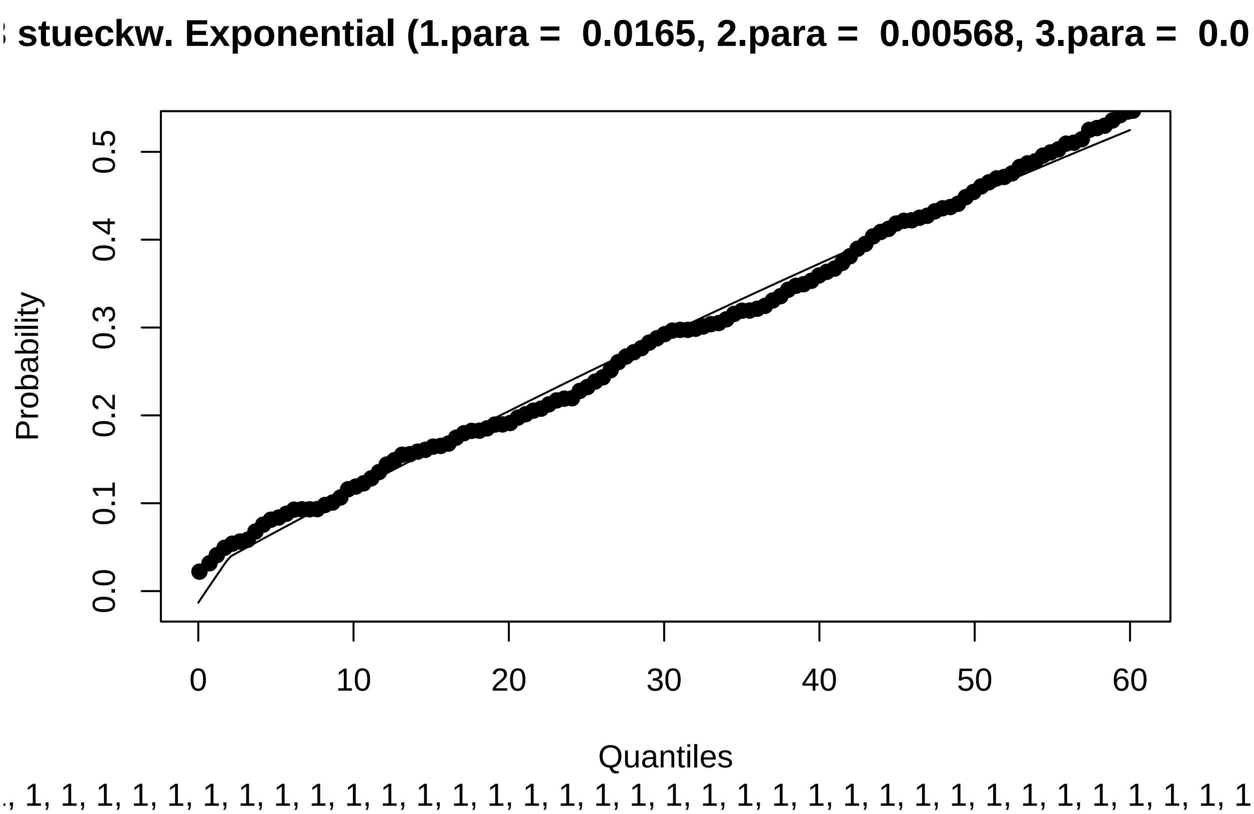

```
best_stuexp3_xb_gleason <- find_best_start_3parameter(p = yb_gleason/100,
                                                       q = xb_gleason,
                                                       max_shape1 = 0.1,
                                                       max_shape2 = 0.1,
                                                       max_scale = 0.1,
                                                       steps_shape1 = 0.01,
                                                       steps_shape2 = 0.01,
                                                       steps_scale = 0.01,
                                                       fitting_function = getstuexp3)
```

```
## Bester Startparameter: 0.06 0.03 0.03
## Bester Fehler: 1.164132e-06
```

```
stuexp3_xb_gleason <- getstuexp3(
                        p = yb_gleason/100,
                        q = xb_gleason,
                        start = best_stuexp3_xb_gleason,
                        show.output = TRUE,
```

```
                         plot = TRUE,
                         wert1 = 2,
                         wert2 = 6
                       )
```

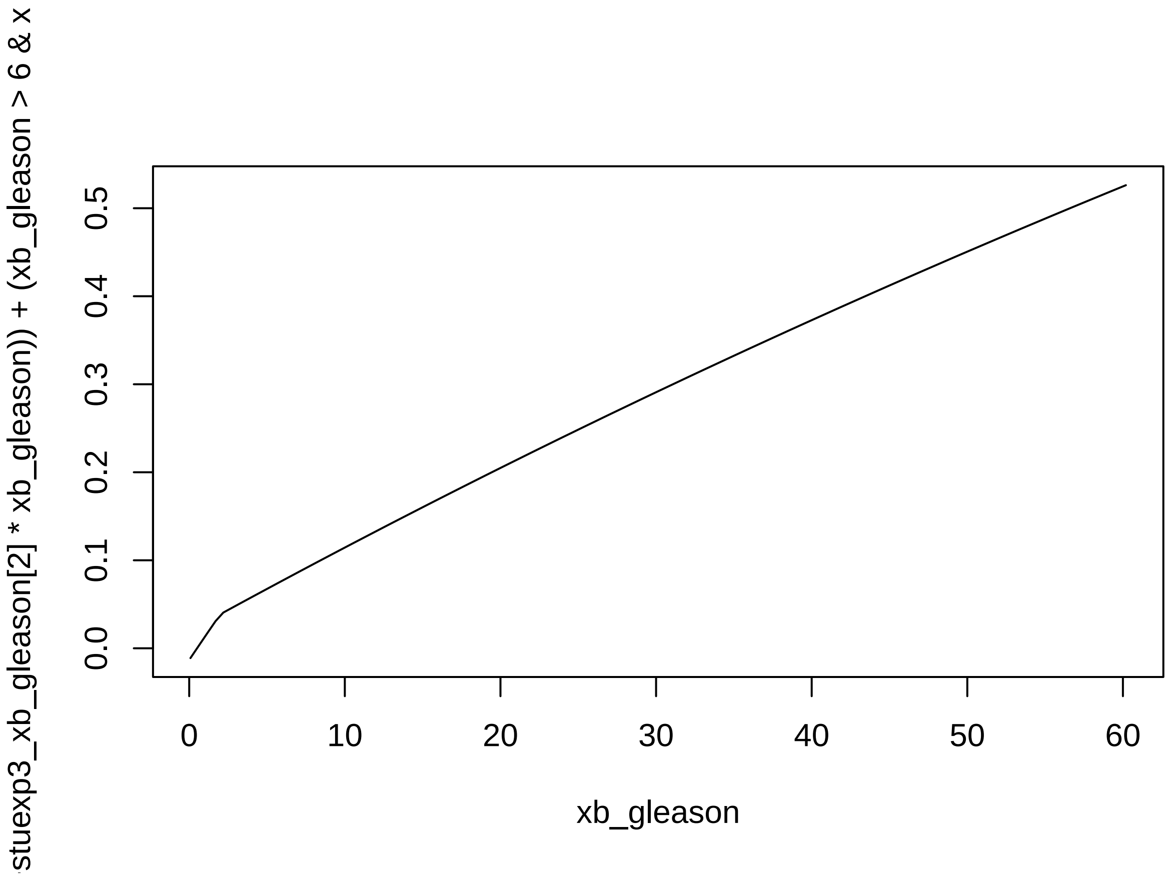
```
## $par
## [1] 0.016525176 0.005683986 0.004065609
##
## $value
## [1] 1.164132e-06
##
## $counts
## function gradient
##       32       32
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```




```
stuexp3_xb_gleason
```

```
##      1.para      2.para      3.para
## 0.016525176 0.005683986 0.004065609
```

```r
plot(xb_gleason,
     ((xb_gleason > 0 & xb_gleason <= 2 ) * (1 - exp(-stuexp3_xb_gleason[1] * xb_gleason)) +
                              (xb_gleason > 2 & xb_gleason <= 6 ) *
              (1 - exp(-stuexp3_xb_gleason[1] * 2))  +
                              (exp(-2 * stuexp3_xb_gleason[2]) - exp(-stuexp3_xb_gleason[2] *
                                                                 xb_gleason)) +
                              (xb_gleason > 6 & xb_gleason <= 65 ) *
              (1 - exp(-stuexp3_xb_gleason[1] * 2)) +
                              (exp(-2 * stuexp3_xb_gleason[2]) - exp(-6 * stuexp3_xb_gleason[2])) +
                              (exp(-6 * stuexp3_xb_gleason[3]) -
                                       exp(-stuexp3_xb_gleason[3] * xb_gleason))),
     type = "l")
```



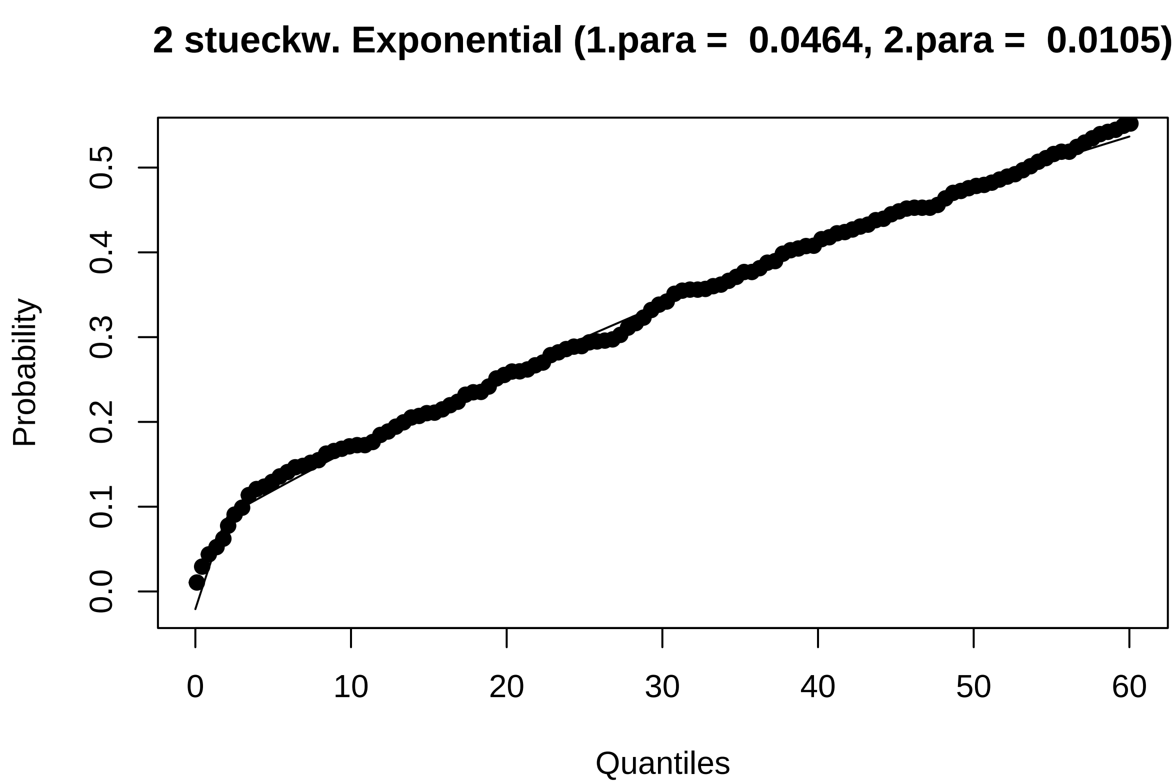
2-Stufige Exponetialverteilung

```r
# 2-Stufige Exponetialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getstuexp2.R")

best_stuexp2_xr_gleason <- find_best_start_2parameter(p = yr_gleason/100,
                                                      q = xr_gleason,
                                                      max_beta = 1,
                                                      max_eta = 0.5,
                                                      steps_beta = 0.1,
                                                      steps_eta = 0.01,
                                                      fitting_function = getstuexp2)
```

```
## Bester Startparameter: 0.2 0.07
## Bester Fehler: 4.676708e-07
```

```r
stuexp2_xr_gleason <- getstuexp2(
                        p = yr_gleason/100,
                        q = xr_gleason,
                        start = best_stuexp2_xr_gleason,
                        show.output = TRUE,
                        plot = TRUE,
                        wert1 = 2
                      )
```

```
## $par
## [1] 0.04639679 0.01054610
##
## $value
## [1] 4.676708e-07
##
## $counts
## function gradient
##       39       39
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

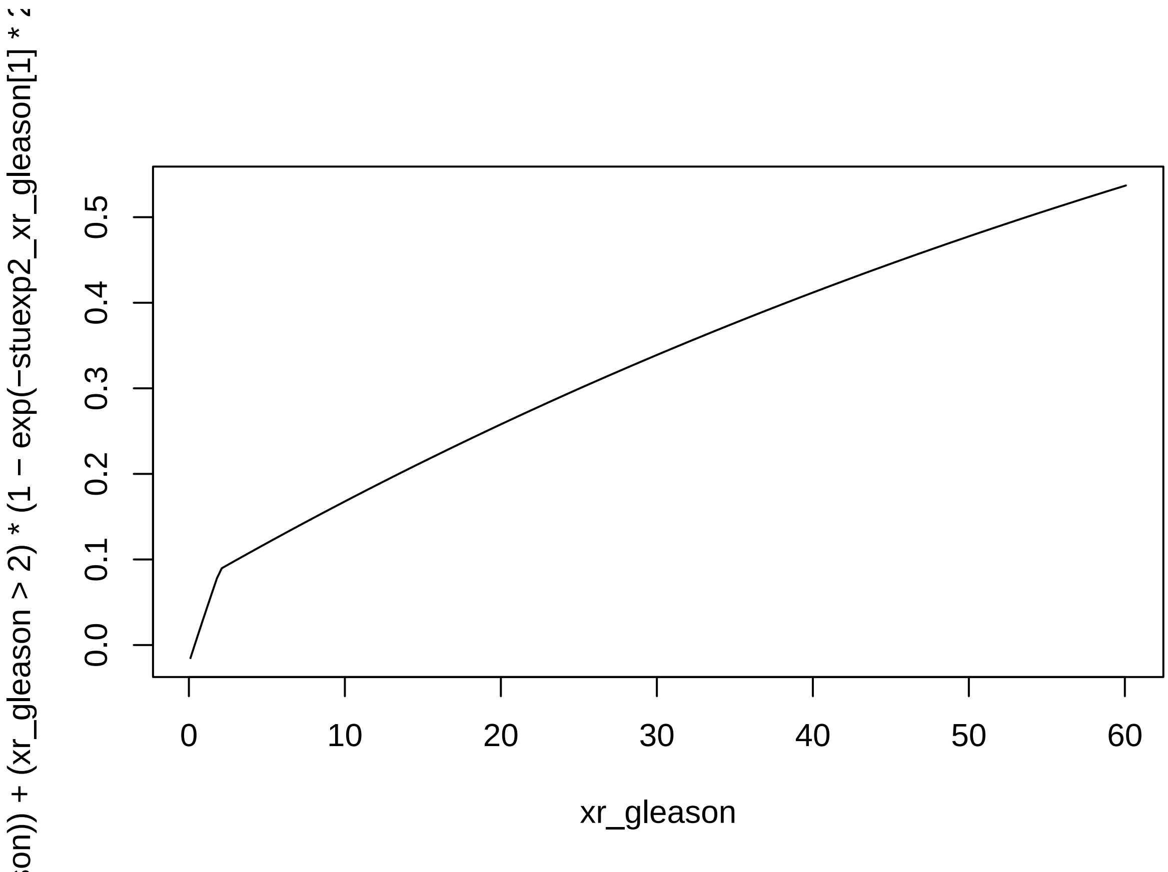


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
stuexp2_xr_gleason
```

```
##     1.para     2.para
## 0.04639679 0.01054610
```

```
plot(xr_gleason,
     ((xr_gleason > 0 & xr_gleason <= 2 ) * (1 - exp(-stuexp2_xr_gleason[1] * xr_gleason)) +
                              (xr_gleason > 2 ) *  (1 - exp(-stuexp2_xr_gleason[1] * 2))  +
                              (exp(-2 * stuexp2_xr_gleason[2]) -
                                       exp(-stuexp2_xr_gleason[2] * xr_gleason))),
     type = "l")
```



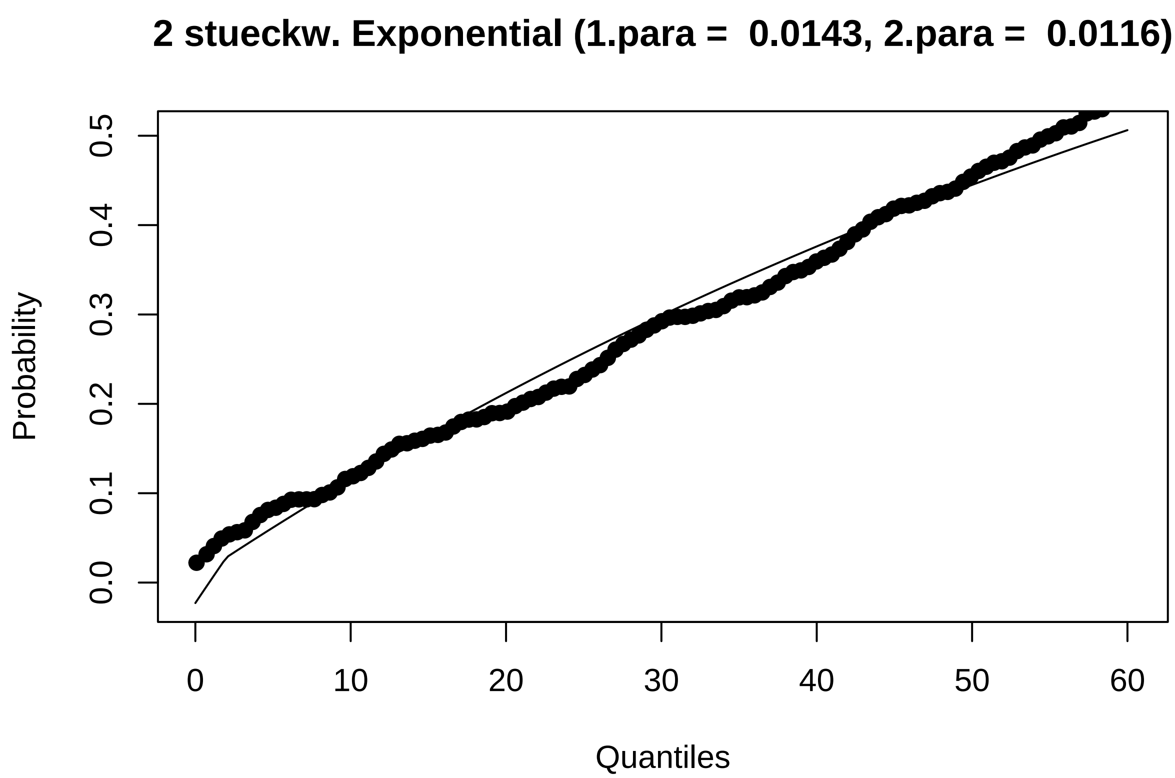

```
best_stuexp2_xb_gleason <- find_best_start_2parameter(p = yb_gleason/100,
                                                     q = xb_gleason,
                                                     max_beta = 1,
                                                     max_eta = 0.5,
                                                     steps_beta = 0.1,
                                                     steps_eta = 0.01,
                                                     fitting_function = getstuexp2)
```

```
## Bester Startparameter: 0.2 0.06
## Bester Fehler: 2.871284e-06
```

```
stuexp2_xb_gleason <- getstuexp2(
                        p = yb_gleason/100,
                        q = xb_gleason,
                        start = best_stuexp2_xb_gleason,
```

```
                      show.output = TRUE,
                      plot = TRUE,
                      wert1 = 2
                    )
```

```
## $par
## [1] 0.01429701 0.01158507
##
## $value
## [1] 2.871284e-06
##
## $counts
## function gradient
##       21       21
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

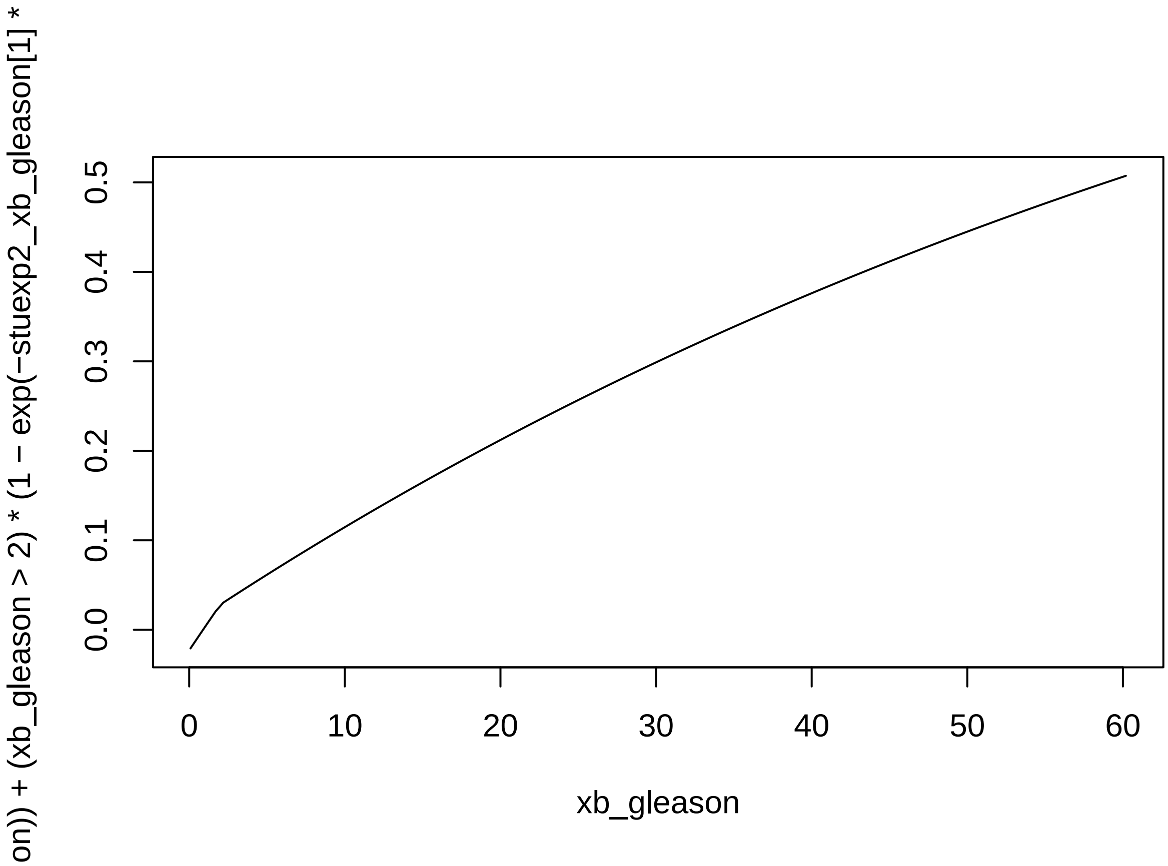


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```
stuexp2_xb_gleason
```

```
##      1.para     2.para
## 0.01429701 0.01158507
```

```r
plot(xb_gleason,
     ((xb_gleason > 0 & xb_gleason <= 2 ) * (1 - exp(-stuexp2_xb_gleason[1] * xb_gleason)) +
                                (xb_gleason > 2 ) *  (1 - exp(-stuexp2_xb_gleason[1] * 2))  +
                                (exp(-2 * stuexp2_xb_gleason[2]) -
                                         exp(-stuexp2_xb_gleason[2] * xb_gleason))),
     type = "l")
```



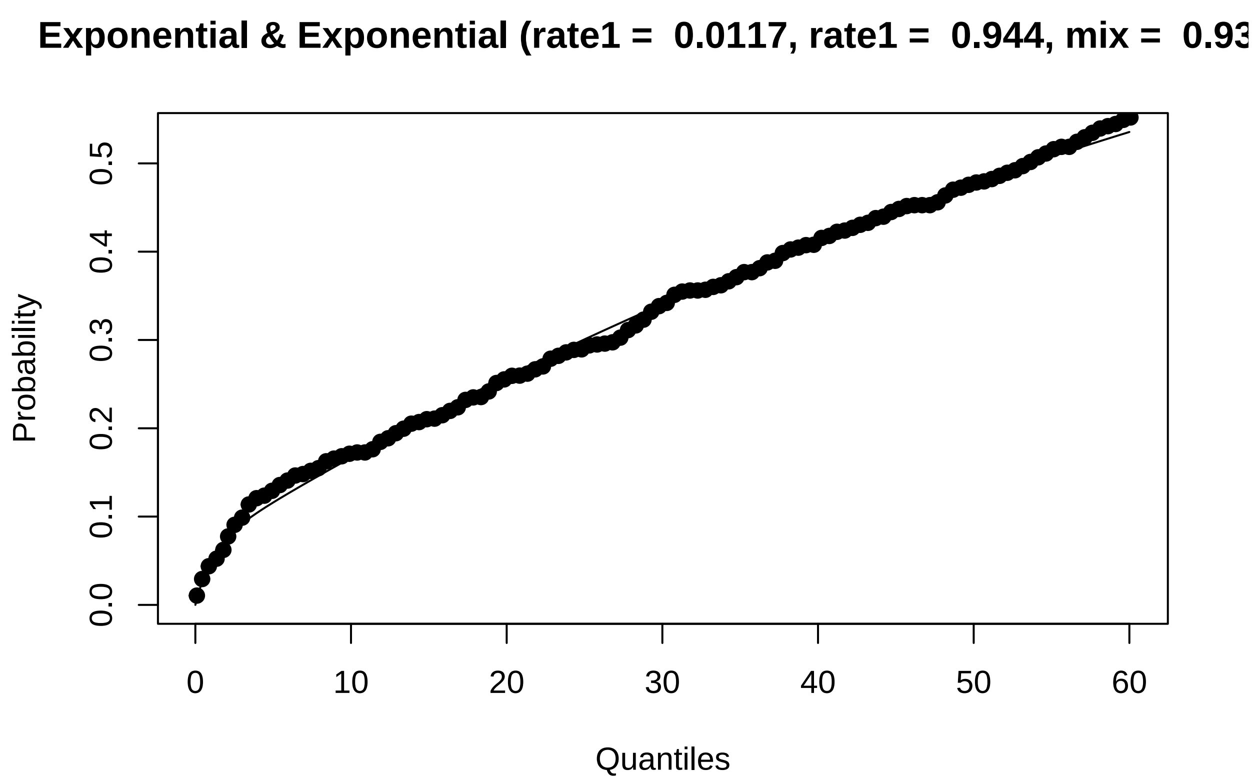
Mischung aus 2 Exponentialverteilungen

```r
# Mischung aus 2 Exponentialverteilungen
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getexex.R")

best_exex_xr_gleason <- find_best_start_3parameter(p = yr_gleason/100,
                                                   q = xr_gleason,
                                                   max_shape1 = 0.7,
                                                   max_shape2 = 0.7,
                                                   max_scale = 1,
                                                   steps_shape1 = 0.1,
                                                   steps_shape2 = 0.1,
                                                   steps_scale = 0.1,
                                                   fitting_function = getexex)
```

```
## Bester Startparameter: 0.2 0.5 0.1
## Bester Fehler: 4.561603e-07
```

```r
exex_xr_gleason <- getexex(
                         p = yr_gleason/100,
                         q = xr_gleason,
                         start = best_exex_xr_gleason,
                         show.output = TRUE,
                         plot = TRUE
                       )
```

```
## $par
## [1] 0.01169027 0.94381688 2.69388592
##
## $value
## [1] 4.561603e-07
##
## $counts
## function gradient
##       53       53
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



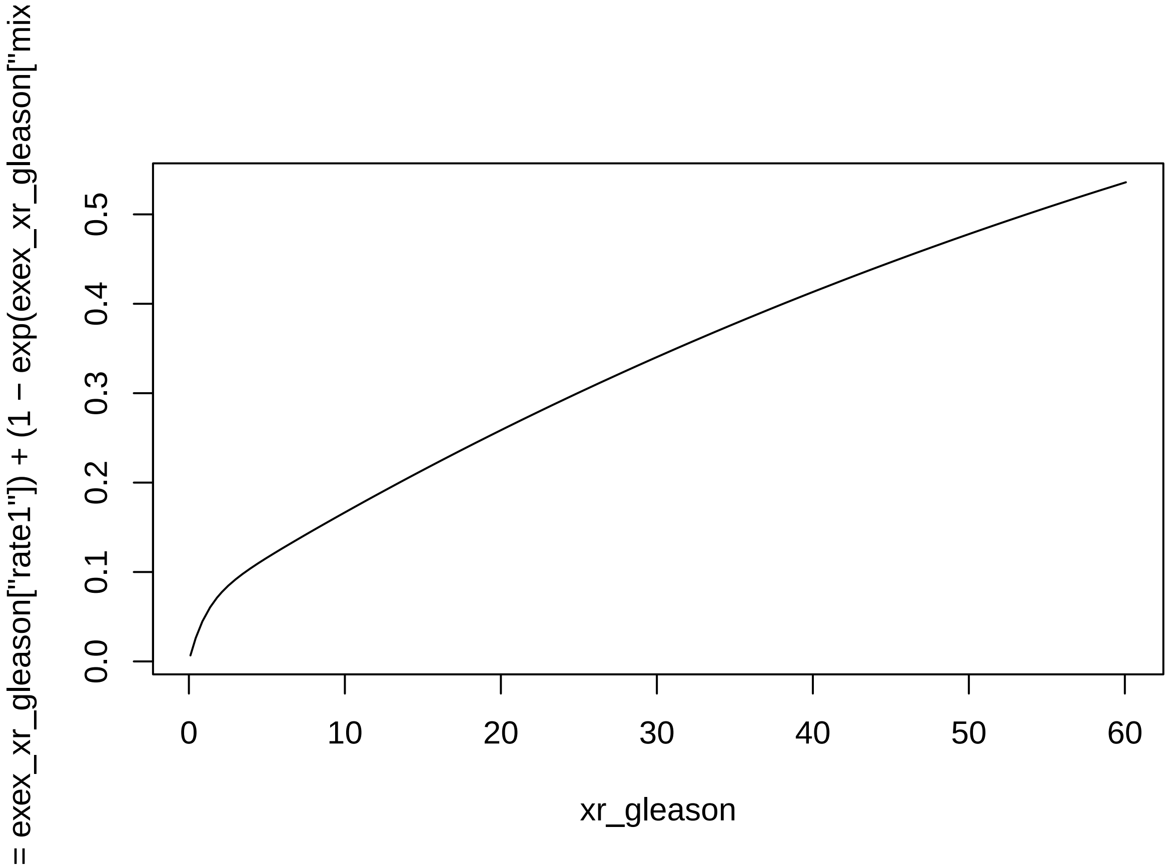
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```r
exex_xr_gleason
```

```
##       rate1       rate2         mix
```

```
## 0.01169027 0.94381688 2.69388592
```

```r
plot(xr_gleason,
     (exp(exex_xr_gleason["mix"]) / ( 1 + exp(exex_xr_gleason["mix"])) *  stats::pexp(q = xr_gleason,
                                                                             rate = exex_xr_gleason["rat
                      (1 - exp(exex_xr_gleason["mix"]) / ( 1 + exp(exex_xr_gleason["mix"]))) *
              stats::pexp(q = xr_gleason,
                          rate = exex_xr_gleason["rate2"])),
     type = "l")
```



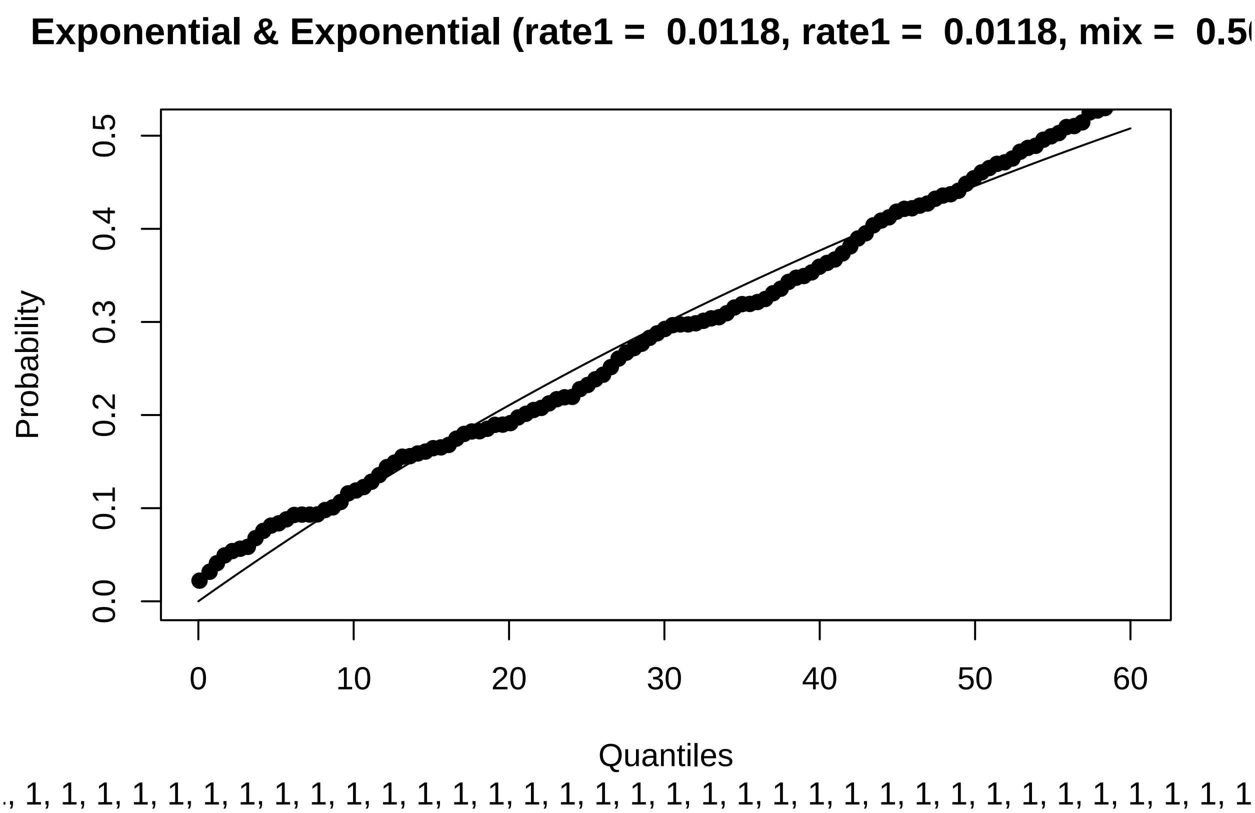

```r
best_exex_xb_gleason <- find_best_start_3parameter(p = yb_gleason/100,
                                                   q = xb_gleason,
                                                   max_shape1 = 0.7,
                                                   max_shape2 = 0.7,
                                                   max_scale = 1,
                                                   steps_shape1 = 0.1,
                                                   steps_shape2 = 0.1,
                                                   steps_scale = 0.1,
                                                   fitting_function = getexex)
```

```
## Bester Startparameter: 0.2 0.1 0.1
## Bester Fehler: 2.694908e-06
```

```r
exex_xb_gleason <- getexex(
                        p = yb_gleason/100,
                        q = xb_gleason,
                        start = best_exex_xb_gleason,
                        show.output = TRUE,
```

```
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.01181548 0.01181853 0.01496963
##
## $value
## [1] 2.694908e-06
##
## $counts
## function gradient
##       33       33
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

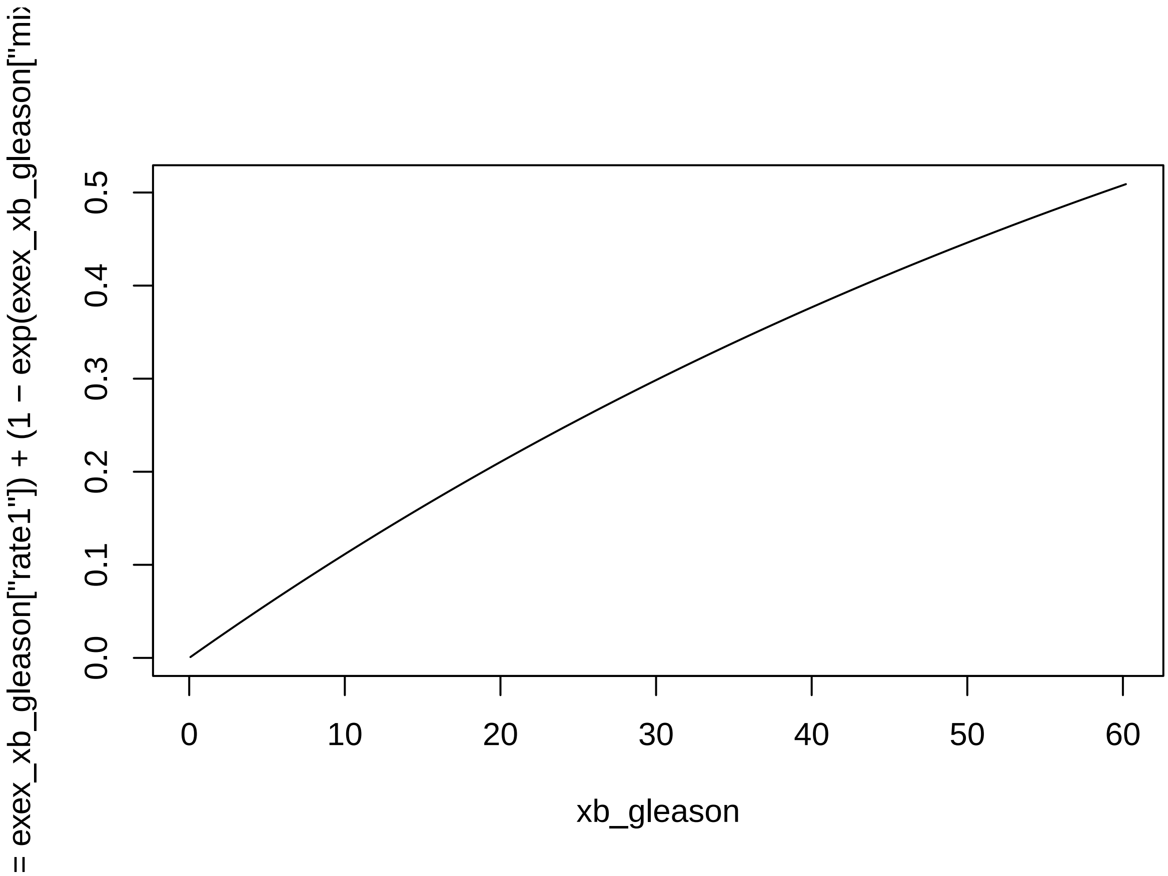



```
exex_xb_gleason
```

```
##       rate1      rate2        mix
## 0.01181548 0.01181853 0.01496963
```

```
plot(xb_gleason,
     (exp(exex_xb_gleason["mix"]) / ( 1 + exp(exex_xb_gleason["mix"])) *  stats::pexp(q = xb_gleason,
                                                                              rate = exex_xb_gleason["rat
```

```
                (1 - exp(exex_xb_gleason["mix"]) / ( 1 + exp(exex_xb_gleason["mix"]))) *
          stats::pexp(q = xb_gleason,
                      rate = exex_xb_gleason["rate2"])),
     type = "l")
```



Ergebnnis

```
getvalue <- function(p, q, best_start, fitting_function){
  if(identical(as.character(substitute(fitting_function)), "getstuexp2")){
    output <- capture.output({
    result <- getstuexp2(p = p, q = q, start = best_start, show.output = TRUE,
                         plot = FALSE, wert1 = 2)
    })
  }
  else if(identical(as.character(substitute(fitting_function)), "getstuexp3")){
          output <- capture.output({
            result <- getstuexp3(
                              p = p, q = q, start = best_start,
                              show.output = TRUE, plot = FALSE, wert1 = 2, wert2 = 6)
          })
  }
  else{
    output <- capture.output({
    result <- fitting_function(p = p, q = q, start = best_start,
```

```
                                show.output = TRUE, plot = FALSE)
    })
  }

  # Berechne den Fehler (gleason_r_1$value) für die aktuellen Startparameter
  error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

  return(error)
}

best_test <- function(p, q, weibull, weib, weibex, weibex2, expweib, stuexp3, stuexp2,
                      exex, start_weibull, start_weib, start_weibex, start_weibex2,
                      start_expweib, start_stuexp3, start_stuexp2, start_exex,
                      group){
  weibull_val <- getvalue(p, q, start_weibull, getweibullpar)
  weib_val <- getvalue(p, q, start_weib, getweibpar)
  weibex_val <- getvalue(p, q, start_weibex, getweibex)
  weibex2_val <- getvalue(p, q, start_weibex2, get2weibex)
  expweib_val <- getvalue(p, q, start_expweib, getexpweib)
  stuexp3_val <- getvalue(p, q, start_stuexp3, getstuexp3)
  stuexp2_val <- getvalue(p, q, start_stuexp2, getstuexp2)
  exex_val <- getvalue(p, q, start_exex, getexex)

  error_distribution_pairs <- list(
    list(weibull_val, "W"),
    list(weib_val, "e.W."),
    list(weibex_val, "M. W&E"),
    list(weibex2_val, "M. e.W&E"),
    list(expweib_val, "e.W o. lambda = 1"),
    list(stuexp3_val, "3 s.E."),
    list(stuexp2_val, "2 s.E."),
    list(exex_val, "M. E&E")
  )

  # Suchen Verteilung mit dem kleinsten Fehler
  best_pair <- error_distribution_pairs[[which.min(sapply(
    error_distribution_pairs, function(pair) pair[[1]]))]]

  # Drucken Sie die Ergebnisse
  cat("Beste Verteilung:", best_pair[[2]], "\n")
  cat("Bester Fehler:", best_pair[[1]], "\n")
  cat("Gruppe: ", group)

  return(c(group, best_pair[[2]],  best_pair[[1]]))
}

best_tavr <- best_test(yb_gleason/100, xb_gleason, gleason_b_1, weibbull_xb_gleason,
                       weibex_xb_gleason, weibex2_xb_gleason, expweib_xb_gleason,
                       stuexp3_xb_gleason, stuexp2_xb_gleason, exex_xb_gleason,
                       best_gleason_b_1, best_weibbull_xb_gleason,
                       best_weibex_xb_gleason, best_weibex2_xb_gleason,
                       best_expweib_xb_gleason, best_stuexp3_xb_gleason,
                       best_stuexp2_xb_gleason, best_exex_xb_gleason, "TAVR")
```

```
## Beste Verteilung: M. W&E
## Bester Fehler: 3.401223e-07
## Gruppe:  TAVR
```

```
best_savr <- best_test(yr_gleason/100, xr_gleason, gleason_r_1, weibbull_xr_gleason,
                       weibex_xr_gleason, weibex2_xr_gleason, expweib_xr_gleason,
                       stuexp3_xr_gleason, stuexp2_xr_gleason, exex_xr_gleason,
                       best_gleason_r_1, best_weibbull_xr_gleason,
                       best_weibex_xr_gleason, best_weibex2_xr_gleason,
                       best_expweib_xr_gleason, best_stuexp3_xr_gleason,
                       best_stuexp2_xr_gleason, best_exex_xr_gleason, "SAVR")
```

```
## Beste Verteilung: M. W&E
## Bester Fehler: 1.537623e-07
## Gruppe:  SAVR
```

```
tab <- matrix(c("CoreValve", "HiRi", "T", best_tavr[1], best_tavr[2],
                best_tavr[3], NA, NA, weibex_xb_gleason[1:2], NA,
                weibex_xb_gleason[3:4],
                "CoreValve", "HiRi", "T",  best_savr[1], best_savr[2],
                best_savr[3], NA, NA, weibex_xr_gleason[1:2], NA,
                weibex_xr_gleason[3:4]),
              ncol=13, byrow=TRUE)

rownames(tab) <- NULL
colnames(tab) <- c('Studie', 'PG', 'EP', 'GR', 'Verteilung', 'SSE', '$\\alpha$',
                   '$\\theta$', '$\\lambda_1$', '$\\lambda_2$', '$\\lambda_3$',
                   '$\\vartheta$', '$\\psi$')

results <- as.data.frame(tab)

# Speichern
write.table(results, "results_gleason.txt", sep = "\t", row.names = FALSE)

# Funktion zur Überprüfung von NA-Werten für Zeichenketten und numerische Werte
is_non_empty <- function(x) {
  return(!is.na(x) & x != "")
}

# Spalten mit mindestens einem nicht-NA-Wert ermitteln
nicht_leere_spalten <- colSums(sapply(results, is_non_empty)) > 0

# Konvertieren Sie die Tabelle in eine Markdown-Tabelle
print(results[, nicht_leere_spalten])
```

```
##      Studie   PG EP   GR Verteilung          SSE     $\\lambda_1$
## 1 CoreValve HiRi  T TAVR     M. W&E 3.401223e-07 1.58316330702727
## 2 CoreValve HiRi  T SAVR     M. W&E 1.537623e-07 1.20513439957757
##       $\\lambda_2$      $\\vartheta$          $\\psi$
## 1 76.3685495805676 0.283167784407591 2.27266255637672
## 2 82.8901131188582 0.523077730768398 2.13757888302752
```